版权注意事项：
1、书籍版权归作者和出版社所有
2、本PDF仅限用于个人获取知识，进行私底下的知识交流
3、PDF获得者不得在互联网上以任何目的进行传播
4、如觉得书籍内容很赞，请购买正版实体书，支持作者
5、请于下载PDF后24小时内删除本PDF。

Broadview®
www.broadview.com.cn



微信小程序开发
零基础入门

张帆　陈思含　编著

- 从易到难、循序渐进，更轻松地掌握微信小程序开发的各种知识。
- 广泛涉及、重点突出，更全面系统地学习小程序开发的各种方法。
- 理论结合实践，更深入地理解小程序的控件和API的使用方法。
- 案例讲解使用工程化思想，真正掌握小程序系统开发的精髓。

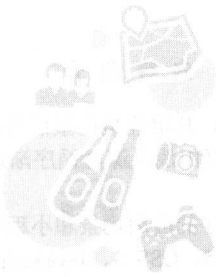
中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

张帆

巨人网络（上海）公司网页开发工程师。喜欢在博客、GitHub 和简书上持续分享各种开发技术和编程实例，有着丰富的项目经验。喜欢新技术和最新的开发方式，喜欢 JavaScript 系的 React.js、Vue.js 等技术，精通网页后台开发、项目管理等。

陈思含

厦门卫星定位应用股份有限公司开发工程师。从事交通领域大数据处理以及 B/S 相关系统的开发工作，精通 Java 领域的相关开发，精通 JavaScript、CSS、HTML 等小程序相关的技术，并且有开发和运营微信公众号的经验。



微信小程序开发零基础入门

张帆　陈思含　编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

微信小程序是一种最新的 App 替代品，其定位是在微信公众号的生态链中，使用微信本身作为应用平台，无须更新和下载，开箱即用。本书由浅入深、循序渐进，使零基础的读者能够熟练掌握使用微信开发小程序应用。

本书分为 8 章，第 1~2 章是关于小程序的应用场景和小程序本身的介绍，包括小程序涉及的技术和 IDE 开发环境的使用。第 3 章是 HTML 5、CSS 3、JavaScript 的语法介绍，然后引申出小程序使用的 WXSS、WXML、JavaScript 的相关语法。第 4 章是小程序开发组件的介绍，每个组件都配备了上手示例，帮助读者加深印象。第 5 章是一个完整的小程序项目的开发步骤。第 6 章是调用微信 API 的介绍，对常见 API 提供了示例编码和分析。第 7 章是一个完整的项目，包括前、后端，通过项目实战让读者上手真正的小程序开发。第 8 章是对小程序最新 API（蓝牙、卡券）等的补充。

本书内容精练、重点突出、实例丰富，是面向广大程序员的编程入门参考书，也是学习小程序开发入门的最佳实践，同时也非常适合大中专院校师生自学，从而掌握最新技术，拓宽知识体系。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

微信小程序开发零基础入门 / 张帆，陈思含编著. —北京：电子工业出版社，2017.5
ISBN 978-7-121-31428-5

Ⅰ. ①微… Ⅱ. ①张… ②陈… Ⅲ. ①移动终端－应用程序－程序设计 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2017）第 085026 号

策划编辑：董 英
责任编辑：徐津平
印 刷：三河市良远印务有限公司
装 订：三河市良远印务有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：787×980 1/16 印张：25.75 字数：576 千字
版 次：2017 年 5 月第 1 版
印 次：2017 年 5 月第 1 次印刷
印 数：3000 册 定价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

前　言

对于商家来说：

- App 打开率越来越低！
- App 开发成本越来越高！
- App 部署难度越来越大！
- App 上线周期越来越长！

对于用户来说：

- 手机存储空间越来越不够用！
- App 越来越用不着！
- 垃圾通知越来越多！
- 安全性越来越差！

这一切，用小程序都能解决！

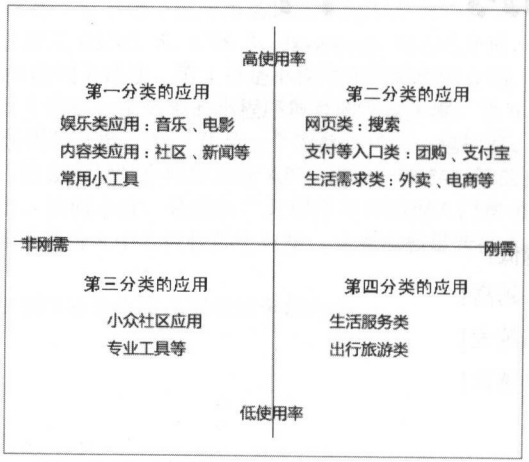
微信小程序通过极低的开发成本，大量简化了相关的开发操作和应用信息获取，结合 HTML 5 和微信的 API，几乎能开发出类似 App 的大多数功能！

迄今为止，已经有大量的应用接入微信小程序端，并迅速攫取了大量的用户，部分应用的使用体验甚至要高于应用本身 App 的体验。通过微信小程序提供的 API，可以方便地实现对手机信息、照片信息及其他媒体信息的获取和操作，而其推送和使用微信账号的登录效果远远方便于原生应用使用第三方插件的体验效果。

对于初学者而言，这是一次技术革新的机会，你可能是开发小白，可能只是一个大学生，可能面对各种移动开发无从选择。本书帮你零基础走进微信小程序开发。作者结合自己多年的后端项目开发和培训经验写作了本书，全面地介绍了微信小程序开发基础、微信小程序通用 API、HTML 5 与 WXML、CSS 3 与 WXSS、JavaScript 等技术，并通过完整的实例介绍了微信小程序开发的步骤和基本架构。

小程序能做什么

大多数内容发布类 App 都可以被小程序所代替，因为微信小程序是一种轻度的云端应用。目前来看，小程序主要应用范围如下：

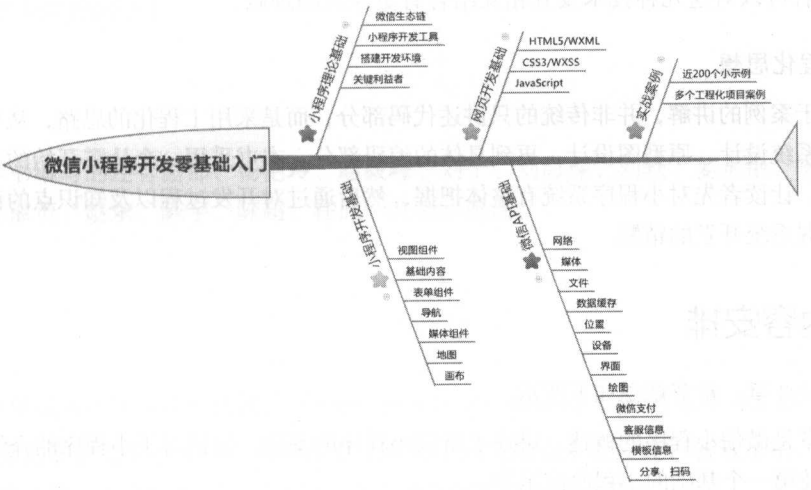


目前已经有很多常用的 App 都开发出了自己的小程序（如下图所示），这样我们就不用再下载这些应用，直接在微信里打开或搜索这些小程序即可。

1	墨迹天气	快递100小助手	知了交通	化妆品点评
2	猫眼电影	食物派Lite	好屋精选	春秋航空特价机票
3	PM2.5空气质量	汕头生活	360好书推荐	翻译e
4	丁香医生	卷皮折扣	雪球炒股	同程旅游
5	荔枝FM	东航旅行	青岛公交查询	桔子创投内参
6	去哪儿酒店	小密圈	贝考英语	轻芒杂志
7	群统计	王者荣耀赛事	新三板股票查询	丽贝小字典
8	亿邦动力lite	京东购物	南宁掌上公交	宝宝微空间
9	家居服务	美团生活	科技日历	股票灯塔
10	手机号归属地查询	腾迅NOW直播	国际品牌价格	分答快问
11	融资计算器	腾讯体育	大脑急转弯	有赞精选
12	全球汇率	腾讯加油卡服务	摩拜单车	印象笔记
13	小鱼收纳	腾讯房产小助手	朝夕万年历	春雨医生
14	IN壁纸	腾讯投票	我的计算器	美柚App
15	喜马拉雅lite	腾讯汽车小助手	火车票机票	e代驾
16	IT查报价	腾讯漫画	菜谱儿	记账e
17	小表情	QQ阅读	饿了么	留学公寓助手
18	每日优鲜	今日头条	美团外卖	多伦多生活黄页
19	企鹅电竞	壹点调查	朝夕日历Pro	南航e行
20	美甲帮	熊猫签证	豆瓣评分	汽车之家
21	查航班	凤凰国际酒店	葡萄生活官方平台	智慧气象服务
22	跨境医药	草料二维码生成器	小小房贷计算器	高校图书馆查询
23	网红助手	毒舌电影社区	飞机票查询预订	爱弹唱
24	叫我修吧	炉石相册	易打卡	一首一首诗
25	小区拼团号	美女汽车	汇率随手算	天天练口语
26	名表价格大全	药代通App	网易新闻	蚂蚁微日记
27	分期计算器	眉眼唇化妆	有书想读	手机查报价
28	恋人清单	掌上探路者生活	小程序时钟	钢琴优课
29	领航员投资挑战赛	头牌网	微寄递	词汇量测试
30	全球职场通讯录	查地铁	家装精品案例	老板管账
31	好奇心日报	博卡名片王	探城美食App	二手车e
32	土巴兔装修效果	懒虫背单词	妈妈网孕育	什么车好

本书对读者的帮助

本书是针对小程序开发的完全零基础入门读物，所以本书包括了小程序的基础、微信 API 的基础、网页开发（HTML 5+CSS 3+JavaScript）的基础，还有完整工程化的项目案例，本书的知识体系如下图所示：



本书的特点

1. 重点突出

全书共分 8 章，采用从易到难、循序渐进的方式进行讲解。内容涉及了微信小程序从开发到实际项目实践的全过程。

2. 涉及广泛

由于程序是相对网页开发的一种技术，为了方便读者学习，本书首先让读者了解 HTML、CSS、JavaScript 等 Web 基础开发技术，并掌握小程序的开发环境和工具。读者在掌握开发环境和工具的基础上，逐渐学习微信小程序特有标签和 API，掌握小程序的系统构建。读者可以边学习边实践，更快掌握微信小程序开发的各种知识。

3. 重点实践

书中每个 API 和控件都有练习实例，并且包含了完整的代码。例程都采用了分步骤实现方法，

这样读者可以很清晰地知道每个技术的具体实现步骤，从而提高学习效率。

4．理论和实践相互结合

由于小程序本身相关技术较多，即使读者理解其控件和 API 对于如何开发一个小程序可能也无从下手。本书通过传统讲解的方式并结合每一个实例，帮助读者解决这个问题。通过对具体项目的实践，读者可以对这几种技术及其相互结合有更深入的理解。

5．工程化思想

本书对于案例的讲解，并非传统的只讲述代码部分，而是采用工程化的思路，从项目需求开始进行分析、系统设计、原型图设计，再到具体的编码部分。本书采用一个从零开始的案例，对系统做整体分析，让读者先对小程序系统有整体把握，然后通过对开发过程以及知识点的详细讲解，使读者真正掌握系统开发的精髓。

本书的内容安排

本书一共 8 章，章节规划如下所示。

第 1~2 章是微信小程序的概述。讲述了微信小程序的基础，包括对于小程序的介绍和开发环境的搭建，以及第一个基础的小程序演示等。

第 3 章是 Web 开发基础知识。讲述了 Web 静态开发的 HTML 5、CSS 3、JavaScript 等知识，并且将其与微信 WXSS、WXML 做了相关的综述对比。

第 4 章和第 6 章是微信小程序标签与 API。讲述了微信小程序开发中的 WXML 标签和微信小程序为开发者提供的 API，并且对于每一个知识点进行了介绍和实例分析。

第 5 章和第 7 章是两个完整的项目实践。通过实践开发两个具有典型代表性的微信小程序应用程序，从零开始，并提供详细开发步骤，介绍了微信小程序应用程序的开发过程，通过对整个流程的介绍，既复习了前面学习的知识，也了解了项目实践的流程和技巧。

第 8 章补充了微信在 2017 年 3 月最新开发的一些 API，包括蓝牙、卡券、收货地址等，由于微信开发对 ES 6 的支持，本章也简单介绍了 ES 6 语法的变化。

本书由浅入深，由理论到实践，尤其适合初级读者逐步学习和完善自己的知识结构。

适合阅读本书的读者

- 希望做微信小程序开发的人员。

- Web 网页开发人员。
- 微信开发人员。
- 微信公众号管理人员。
- 想将原生 App 制作为微信小程序的人员。
- 想降低 App 开发成本的初创企业。
- 大中专院校的学生。

致谢

参与本书编写的还有谢郁、杨艳玲、赵毅峰、刘宁、刘海琛、刘轶、姜永艳、李玉莉、杨超、林江闽、薛淑英、赵东、陈宇、陈超，在此一并表示感谢。

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误**：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与作者交流**：在页面下方【读者评论】处留下您的疑问或观点，与作者和其他读者一同学习交流。



页面入口：http://www.broadview.com.cn/31428

目 录

1 微信小程序开发 ……1

1.1 初识微信小程序开发……1

1.1.1 什么是微信公众号……1

1.1.2 什么是微信小程序……3

1.1.3 微信小程序的应用领域……4

1.1.4 怎样学好微信小程序……5

1.1.5 微信小程序的前景……6

1.2 小程序开发需要的技术……8

1.2.1 JavaScript 技术……8

1.2.2 CSS 技术……9

1.2.3 HTML（XML）技术……10

1.3 小程序开发所使用的技术……11

1.3.1 开发文件结构……12

1.3.2 配置程序……12

1.3.3 逻辑层……16

1.3.4 WXML……20

1.3.5 WXSS……25

1.4 搭建小程序开发环境……27

1.4.1 IDE 下载……27

1.4.2 Windows 系统的 IDE 安装……29

1.5 上手实践：第一个 Hello WX 程序……31

1.6 小程序新能力……34

1.6.1 个人开发者申请小程序……35

1.6.2 小程序的基本信息管理……37

1.6.3 已有公众号绑定小程序……40

1.6.4 小程序的分享和跳转新模式……41
1.7 小结……43
1.8 实践与练习……43

2 熟悉小程序开发工具……44

2.1 熟悉微信开发者工具……44
2.1.1 微信开发 IDE 简介……44
2.1.2 IDE 调试窗口……45
2.1.3 小程序的测试与重启……49
2.1.4 小程序的调试……50
2.1.5 小程序的控制台……53
2.1.6 IDE 项目窗口……55
2.1.7 项目的关闭与新建……56
2.1.8 IDE 项目编辑窗口……57
2.2 小结……59
2.3 实践与练习……59

3 小程序开发基础……60

3.1 HTML 入门……60
3.1.1 HTML 简介……60
3.1.2 HTML 的用途与结构……61
3.2 HTML 基础……62
3.2.1 HTML 语法基础……62
3.2.2 HTML 元素……63
3.2.3 HTML 属性……77
3.2.4 WXML 与 HTML……78
3.3 CSS 入门……79
3.4 CSS 基础……79
3.4.1 CSS 基础语法……80
3.4.2 CSS 样式……82
3.4.3 CSS 框模型……95
3.4.4 CSS 定位……101
3.4.5 CSS 与 WXSS……106
3.5 JavaScript 入门……106

3.5.1 JavaScript 简介……106
3.5.2 JavaScript 前景……107
3.6 JavaScript 基础……108
3.6.1 JavaScript 注释……108
3.6.2 JavaScript 变量……108
3.6.3 JavaScript 数据类型……112
3.6.4 JavaScript 对象……118
3.6.5 JavaScript 函数……122
3.6.6 JavaScript 运算符……127
3.6.7 JavaScript 逻辑控制语句……128
3.7 小结……142
3.8 实践与练习……143
4 小程序开发组件……144
4.1 视图组件……144
4.1.1 视图容器（view）组件……144
4.1.2 可滚动视图区域（scroll-view）……147
4.1.3 滑动界面（swiper）……147
4.2 基础内容……149
4.2.1 小图标（icon）……149
4.2.2 文本组件（text）……150
4.2.3 进度条组件（progress）……152
4.3 表单组件……153
4.3.1 按钮组件（button）……153
4.3.2 多选择框（checkbox）……156
4.3.3 表单 form……159
4.3.4 输入框组件（input）……160
4.3.5 label 元素……163
4.3.6 滚动选择器 picker……163
4.3.7 单选控制器（radio）……166
4.3.8 滑动选择器（slider）……168
4.3.9 开关选择器（switch）……170
4.3.10 文本框控件（textarea）……172
4.4 导航……174

4.5 媒体组件 ····· 176
4.5.1 音频组件（audio） ····· 177
4.5.2 上手实践：音乐播放器 ····· 177
4.5.3 图片控件（image） ····· 180
4.5.4 上手实践：相册 ····· 180
4.5.5 视频控件（video） ····· 184
4.5.6 上手实践：带弹幕的视频播放器 ····· 184
4.6 地图 ····· 187
4.6.1 地图 ····· 187
4.6.2 上手实践：微信中的腾讯地图定位 ····· 188
4.7 画布 ····· 189
4.7.1 画布控件（canvas） ····· 190
4.7.2 上手实践：使用 canvas 实现画布的绘制 ····· 190
4.8 小结 ····· 192
4.9 实践与练习 ····· 192

5 项目实战 1——日记本 ····· 193

5.1 需求分析 ····· 193
5.1.1 需求确定 ····· 193
5.1.2 功能确定 ····· 194
5.1.3 系统设计 ····· 194
5.1.4 原型设计 ····· 197
5.2 具体编码 ····· 199
5.2.1 项目初始化 ····· 199
5.2.2 测试环境搭建 ····· 201
5.2.3 用户系统编码 ····· 203
5.2.4 日记系统编码 ····· 218
5.3 小结 ····· 233
5.4 实践与练习 ····· 235

6 微信提供的 API 解析 ····· 236

6.1 网络 ····· 236
6.1.1 发起请求 ····· 236
6.1.2 上传与下载 ····· 238

6.1.3 WebSocket ····· 240
6.2 媒体 ····· 243
6.2.1 图片 ····· 243
6.2.2 录音 ····· 245
6.2.3 语音控制 ····· 246
6.2.4 音乐控制 ····· 248
6.2.5 视频控制 ····· 252
6.2.6 上手实践：获取本地视频和播放控制 ····· 254
6.3 文件 ····· 256
6.4 数据缓存 ····· 260
6.5 位置 ····· 264
6.5.1 获取位置 ····· 264
6.5.2 查看位置 ····· 265
6.5.3 地图组件控制 ····· 266
6.5.4 上手实践：地图控制和查看 ····· 267
6.6 设备 ····· 269
6.6.1 系统信息 ····· 269
6.6.2 网络状态 ····· 271
6.6.3 重力感应 ····· 271
6.6.4 罗盘 ····· 272
6.6.5 拨打电话 ····· 272
6.6.6 扫码 ····· 272
6.6.7 上手实践：获取所有手机信息和网络状态 ····· 273
6.7 界面 ····· 274
6.7.1 交互反馈 ····· 274
6.7.2 设置导航条 ····· 277
6.7.3 导航 ····· 278
6.7.4 动画 ····· 280
6.7.5 下拉刷新 ····· 284
6.7.6 上手实践：带有控制的动画 ····· 285
6.8 绘图 ····· 287
6.8.1 在 canvas 上画图 ····· 289
6.8.2 canvas 坐标系 ····· 290

6.8.3 渐变 ····· 291
6.8.4 上手实践：使用 canvas 绘制笑脸 ····· 294
6.9 开放接口 ····· 296
6.9.1 登录 ····· 296
6.9.2 用户信息 ····· 301
6.9.3 微信支付 ····· 302
6.9.4 模板消息 ····· 303
6.9.5 客服消息 ····· 307
6.9.6 分享 ····· 315
6.9.7 获取二维码 ····· 316
6.10 小结 ····· 316
6.11 实践与练习 ····· 317
7 项目实战 2——在线阅读 ····· 318
7.1 项目准备工作 ····· 318
7.1.1 项目需求分析 ····· 318
7.1.2 项目系统设计 ····· 319
7.1.3 设计 1：用户模块 ····· 319
7.1.4 设计 2：小说阅读模块 ····· 321
7.1.5 项目使用到的技术 ····· 321
7.2 项目服务器端编码 ····· 322
7.2.1 服务器端介绍 ····· 322
7.2.2 Laravel 框架的安装与配置 ····· 322
7.2.3 服务器功能路由划分 ····· 324
7.2.4 服务器数据库搭建 ····· 325
7.2.5 服务器的配置 ····· 327
7.2.6 服务器的项目结构 ····· 328
7.2.7 服务器端登录页面的代码实现 ····· 328
7.2.8 服务器端管理主页的代码实现 ····· 330
7.2.9 服务器端管理用户的代码实现 ····· 332
7.2.10 小程序的接口文档 ····· 332
7.2.11 后台管理页面搭建 ····· 339
7.2.12 服务器端的测试 ····· 347
7.3 项目微信小程序端编码 ····· 352

7.3.1 微信小程序页面设计……353
7.3.2 搭建微信小程序项目……353
7.3.3 微信小程序“首页”菜单的代码实现……355
7.3.4 微信小程序“用户历史”菜单的代码实现……357
7.3.5 微信小程序“用户注册”页面的代码实现……361
7.3.6 微信小程序“小说阅读”页面的代码实现……367
7.4 小结……371
7.5 实践与练习……371

8 小程序新增 API……372

8.1 蓝牙 API……372
8.1.1 获取蓝牙适配器……372
8.1.2 关闭蓝牙模块……374
8.1.3 获取本机蓝牙状态……375
8.1.4 搜索蓝牙设备……377
8.1.5 上手实践：蓝牙管理器……386
8.2 其他新增 API……389
8.2.1 获取收货地址……390
8.2.2 打开小程序设置界面……391
8.2.3 操作剪贴板……392
8.2.4 支持微信卡券……393
8.2.5 监听网络变化……395
8.3 ES 6 的支持……396
8.4 小结……398
8.5 实践与练习……398



微信小程序开发

微信是腾讯公司于2011年1月21日推出的一个为智能终端提供即时通信服务的免费应用程序，支持跨运营商、跨操作系统平台通过网络快速发送免费（需消耗少量网络流量）语音短信、视频、图片和文字，同时，也可以使用基于位置的社交插件“摇一摇”“漂流瓶”“朋友圈”、公众平台、语音记事本等服务插件。

现在微信已经是亚洲地区最大用户群体的移动即时通信软件，这个平台已经远远不止社交如此简单，它已经包含了生活、娱乐、支付、媒体等服务，甚至提供了新式的应用服务入口，而 2017年1月正式面世的小程序，便是最新的应用技术，也昭示着微信想做一个 App 整合平台的野心。

本章涉及的主要知识点如下。

- 微信公众号介绍：介绍微信公众号的特点和优势。
- 微信小程序介绍：介绍微信的小程序功能，并且对此功能进行分析。
- 应用领域：分析小程序的应用场景和适合拥抱小程序的应用类型。
- 优势和特点：通过对小程序特性的剖析，找到小程序的适用范围和有优势的使用场景，并且对其将来的发展趋势做出描述。

1.1 初识微信小程序开发

本节首先介绍了什么是微信小程序应用，以及为什么要使用微信小程序进行开发。理解这些概念是学习微信小程序开发的基础，同时也是为什么选择微信小程序的必备知识。了解了微信小程序的强大，才能增强学习兴趣与乐趣。

1.1.1 什么是微信公众号

微信公众号是开发者或商家在微信公众平台上申请的应用账号，该账号与 QQ 账号互通，通过

公众号，商家可在微信平台上实现和特定群体的文字、图片、语音、视频的全方位沟通、互动，形成了一种主流的线上线下微信互动营销方式。

线上线下微信互动营销的代表微部落，率先提出标准的行业通用模板和深度定制的微信平台开发理念相结合，形成了线上线下微信互动营销的开放应用平台。

微信公众号主要是面向个人、政府、媒体、企业等机构推出的合作推广业务。在这里可以通过微信渠道将品牌推广给数以亿计的微信用户，减少宣传成本，提高品牌知名度，打造更具影响力的品牌形象。微信公众号有三类：

- 服务号。
- 订阅号。
- 企业号。

对于这三种账号的官方介绍，如图 1-1 所示。

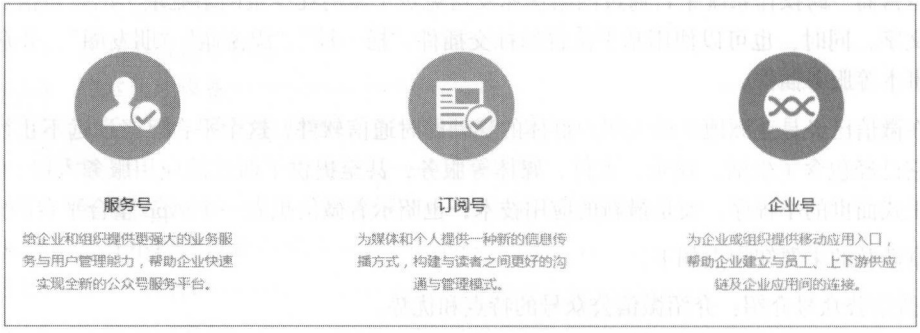


图 1-1　种类介绍

请通过电脑登录微信公众平台官网（http://mp.weixin.qq.com/），注册微信公众号，账号可以是个人用户或者企业用户。个人用户存在一些限制，企业用户需要进行企业资质的审核，但是无论如何都需要使用个人身份证件或者企业资料进行微信公众号审核，没有通过审核的公众号是没有任何用处的。

注册结束之后，登录此账号，会提示绑定管理者的个人微信，在第二次登录的时候会提示扫描登录。使用手机上管理者的微信单击“登录”按钮，网页也会自动跳转到管理界面，一个微信公众号就申请成功了，进入的管理界面如图 1-2 所示。

在这个管理界面里可以选择不同的开发模式，选择开发者模式后可以对接口和应用行为进行管理，对于微信小程序的申请和开发就在这个管理界面中，如图 1-3 所示。

注意：为什么要申请公众号呢？因为小程序就是借助于微信公众号进行发布的。



图 1-2 管理界面



图 1-3 小程序列表

1.1.2 什么是微信小程序

2016 年 9 月 21 日深夜，微信官方发放了一份关于微信小程序的内测邀请函，2016 年年初张小龙提出的应用号终于面世了，这意味着微信又一新的布局的展现。

根据微信官方的说法，小程序是一种不需要下载安装即可使用的应用，它实现了应用“触手可及”的梦想，用户扫一扫或者搜一下即可打开应用。也体现了“用完即走”的理念，用户不用担心是否安装太多应用的问题。应用将无处不在，随时可用，但又无须安装卸载。

小程序依托于微信平台，由于微信多平台的特性，所以小程序自然也会跟随微信进入多平台，实现简单且实用的跨平台体系，有可能成为一个巨大的 App 生态圈。

简单来说，微信小程序是一种基于微信生态圈的 WebApp 式应用。在提供面向用户友好的基础 UI 和交互体验的同时，也定制了统一的开发规范，并且为开发者提供基于微信的表单、导航、地图、媒体和位置等开发组件，这使得开发小程序的过程变得更加便利。

通过对于传统开发方式的打包，让开发者能够非常迅速地在微信的 WebView 里构建一个基于 HTML 5 和 JavaScript 的应用。微信还开放了登录和微信支付等接口，让这个小程序可以充分利用微信这一个平台的资源，对于 Socket 的支持，也使得小程序本身不会太过受限于微信提供的功能，极大地提高了开发的便利度和拓展性。

1.1.3 微信小程序的应用领域

2016 年 1 月，“微信之父”张小龙就曾宣布微信公众号将推出应用号，他当时的观点是，“很多人做微信公众号，而非 App，因为 App 推广成本太高。这导致公众号现在更像是媒体化的平台。但我们本意不是传播内容，我们要提供服务，所以推出了服务号，但是服务号也没达到预期效果。我们在讨论一个新形态，叫应用号。用户关注应用号，就像安装一个 App 一样。”

所以大多数的内容发布类App都可以被小程序所代替，因为微信小程序是一种轻度的云端应用。

由于云和网络的发展，整个 App 的生态不再局限于本地一些应用的模式，而是逐渐进化成一种基于云端的体验，由于 HTML 5 的出现和服务器运载量以及带宽的高速发展，还有市场自由竞争的发展，使得用户不再被限制于某一个 App 提供的某项服务，但是手机的存储能力就成为了一个发展瓶颈。

用户不能在一个内存时常会满的终端上无止境地下载不同的 App 进行服务的体验，也不能时刻保证其在 Wi-Fi 环境下，而使用流量下载几十兆甚至上百兆，却可能只使用一两次的 App 对于用户而言，无疑是天方夜谭。

对于开发者而言，一个良好的跨平台开发环境非常重要。尤其是对于个人或者小团队的开发者而言，快速的市场反应能力远比开发出一个完美的 App 更值得追求，这样，产品才可以选择性地针对市场进行高速迭代。

并不是所有的服务都适合接入小程序，但大部分个人开发者和创业者团队的服务都适合接入其中，在微信的云端发布产品，同时也可以在云端缓存数据，这对于初创团队而言，大量减少了人力成本和运维成本。

适合用于微信小程序的应用分类如图 1-4 所示。



图 1-4　应用分类

对图 1-4 的分类进行一些分析：

- 第一分类的应用是不应该接入小程序的。这个分类都是互联网巨头，贸然接入可能会造成自我生态链的断裂，甚至于数据的安全性也不可以保证，且这个分类的应用需要更好的客户体验和更好的数据加密，所以是不适合接入小程序的。
- 第二分类的应用非常适合小程序的开发。这个分类主要是信息服务类的产品，涉及教育、医疗、家政、求职招聘、二手买卖、旅游、票务、金融理财等，属于非高频的专用信息服务，通过和微信的结合，可以为用户提供更加便捷的登录和预订服务，而不用常驻手机内存，即用即开。
- 第三分类的应用是比较尴尬的分类应用，可以尝试分析是否符合公司的具体生态圈和用户体验推广等，具体情况具体分析。当然，如果能用于打造创业团队，用于试错 MVP 产品减少试错成本，提升成功概率，也是值得去尝试开发的。
- 第四分类的应用比较适用于专业开发，视其特点的产品和分类进行是否介入小程序，因为对于小程序而言，其接口受限制于微信本身的 API，所以对于有些专业的小程序应用可能并不能适用于或者是满足于需求，所以需要视情况而定。

综上，小程序的适用范围是广泛而普遍的，非常适用于中国的创业团队打造一个 MVP，进行市场上的试错，并且可以在微信生态圈里减少了对于前期大量 App Store 的推广费用，并且能获得第一批初始用户。所以，这种快速而灵活的方式正是一个初创团队所需要的，而紧跟科技的脚步，能在一个现有产品上不断快速迭代开发的产品和开发模式才是最适合如今的互联网创业的。

注意：技术和市场是同样重要的，但是对于一个初创团队而言，最重要的并不是做一个足够完美的产品，而是做出一个 Demo 式的产品，然后推向用户，一点点完善，最后成为一个“完美”的产品。

1.1.4 怎样学好微信小程序

微信小程序虽然已经正式发布，但是版本还处于一个不稳定的环境下，所以较 Java、C++这样成熟的技术而言可能会变动较大。所以，要学好这样一门新技术，可能需要下面的一些能力：

1. 基本的技术基础

（1）JavaScript（简称 JS）基础。微信小程序基于 JS，并且支持原生 JS 进行逻辑开发，所以有一定的 JS 基础是非常重要的。

（2）CSS 基础。虽然用于布局样式的 WXSS 并不是单纯的 CSS，但是也是基于 CSS 技术的一种封装，所以有一定的 CSS 基础对于学习 WXSS 也是必要的。

（3）HTML（XML）基础。用于布局的 WXML 同样也是对于 HTML（XML）的一种封装应用，

所以 HTML 基础对于学习也是有一定意义的。

2. 基本的编程思维

（1）熟悉基本的编程思想，拥有写网页或者是 App 的思维模式。

（2）拥有不错的逻辑思维能力，明确需求达到的目标。

3. 解决问题的能力

（1）熟练应用 Google、Baidu 等搜索引擎，找到相应的解决办法。

（2）学会使用 GitHub、StackOverflow 等社区找到解决方案。

4. 自学能力

能够不断自主学习相应的知识和计算机的最新技术，这也是在编程学习中最重要的一点。

其实小程序的开发还是很简单和便捷的，但是这并不意味着它是一个简单的程序，腾讯将服务和接口打包开放给了开发者，这使得开发人员的工作和难度大大减少了，但是正如任何一个入门简单的行业和技术一样，学习曲线都是冗长的。

所以想要实现在一个技术行业的不断成长，总结和学习是非常重要的。

注意：本书不会太多地涉及 JS、HTML 等技术，只是一个粗浅的学习过程，并不是从原理开始讲起，只会涉及一些简单的、我们需要使用的技术。剩下的需要读者不断地自主学习。

1.1.5 微信小程序的前景

根据微信团队的说法，现阶段的小程序可以借助微信联合登录，和开发者已有的 App 后台的用户数据进行打通，但不会支持小程序和 App 的直接跳转。

1. 优势

总体来看，腾讯为小程序的开发者开放多种服务及支撑能力，主要包括以下功能点。

- 视图容器：视图（View）、滚动视图、Swiper。
- 基础内容：图标、文本、进度条。
- 表单组件：按钮、表单等。
- 操作反馈。
- 导航。
- 媒体组件：音频、图片、视频。
- 地图位置服务。

- 画布。
- 文件操作能力。
- 网络：上传下载能力、WebSocket。
- 数据：数据缓存能力。
- 位置：获取位置、查看位置。
- 设备：网络状态、系统信息、重力感应、罗盘。
- 界面：设置导航条、导航、动画、绘图等。
- 开放接口：登录，包括签名加密、用户信息、微信支付、模板消息。

也就是说，可以通过调用已有的接口去实现很多原本只能通过 App 实现的功能，大大简化了开发成本，提高了开发速度。

按照 App 的发展规律和需求而言，小程序这样的应用模式的出现，无疑是很美好的。对开发者来说，微信小程序提供了一整套跨平台方式，这使得用户不用费尽心思开发多平台的 App，尤其是屏幕适配问题不用考虑各种奇形怪状的机型和屏幕分辨率了，对于前端工程师而言更是一件利器。

这其实并非一个新的技术或者是跨时代的一次革命，但是对于微信推进的这次云端 App 而言，无疑是值得去跟进学习的，其可能并非代表着现在，更多的是将来的一种 App 的布局和发展趋势。

还有一点很重要，因为所有的逻辑和页面其实都是存于服务器上的，无须用户的更新和版本推送，不同版本的推送其实就只是一次服务器的版本更新，出现 bug 也可以随时地回滚或者修复版本。整个流程如图 1-5 所示。

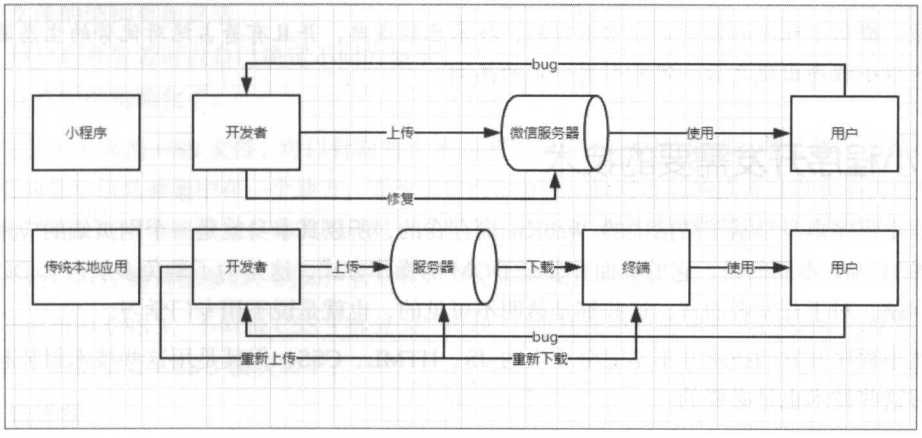


图 1-5 具体流程

对于用户而言，微信小程序带来的更是便捷和手机存储空间的释放，无须安装 App，无须无止境地更新和同步数据，甚至于将登录系统都整合为一个微信账户实现的自动登录。

此处其实能类比微信的服务号业务，比如滴滴出行的服务号，用户打车无须安装滴滴出行 App，而是直接打开滴滴的应用号，单击“我要打车”按钮，输入目的地，就可以实现叫车的业务逻辑。滴滴的应用号如图 1-6 所示。

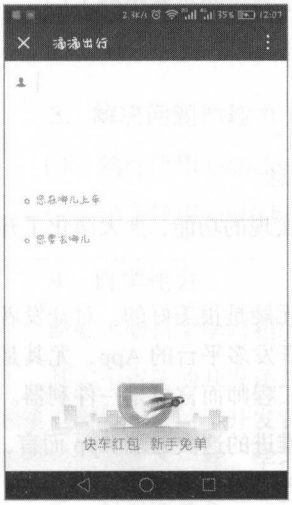


图 1-6 滴滴的应用号

而微信的小程序可以更好地实现这一业务逻辑，并且极大地提高整体的用户体验和应用的性能，给用户提供更好的服务体验。同时，包括外卖、团购等业务将来也可以选择使用小程序来解决方案。无须用户下载，却能给予用户原生或者是接近于原生 App 的体验，通过统一的微信接口解决一切，给用户最佳体验。

2. 不足

微信小程序的未来也不一定一帆风顺，虽然微信小程序提供了大量方便的开发模式和平台，但对于有些大公司的主流大众产品，从利益的角度来说，利用微信的入口和平台流量，就会受制于微信和腾讯本身，这是值得衡量的。

还有一个问题，微信暂时没有解决多任务运行的问题。也就是说，如果正在使用一个应用的时候，突然有朋友给你发送了一条信息，当你去回复信息后，再打开那个小程序可能需要重新刷新应用。

也就是说，无法达到操作系统一样的效果，所以应该分应用、分场景、分人群进行不同类型的程序开发，而不是盲目地拥抱小程序。但是无疑，它的发展前景是非常不错的。

注意：微信小程序的前景是非常不错的，技术已经成熟，并且有着占绝对优势的生态圈和服务平台，所以小程序出现之后一定能引发一定的热潮。

1.2 小程序开发需要的技术

微信小程序本身是基于微信中的 WebKit 而存在的，所以其本身就是一个网页端的应用。但是与传统的 HTML 不同的是，它的页面并非是 DOM 的操作方式，这是为了避免多层嵌套 DOM 速度过慢的问题，对于开发者而言，这些都是透明不可见的，也就是说不用专门学习。

虽然小程序开发的技术并不是完全原生的 JS、HTML、CSS，但其是用这些技术封装起来的，所以学习这些技术也是必要的。

1.2.1 JavaScript 技术

JavaScript 是世界上最流行的脚本语言，它属于 Web 语言，它适用于 PC、笔记本电脑、平板电

脑和移动电话。JavaScript 的设计目的是增加 HTML 页面的交互性。

许多 HTML 开发者都不是程序员，但 JavaScript 拥有非常简单的语法，所以几乎每个人都有能力将小的 JavaScript 片段添加到网页中。

JavaScript 本身是一种高级编程语言，通过解释执行，是面向对象（基于原型）的直译语言。它已经由 ECMA（欧洲电脑制造商协会）通过 ECMAScript 实现语言的标准化。它被世界上的绝大多数网站所使用，也被世界主流浏览器（Chrome、IE、FireFox、Safari、Opera）支持。

在客户端，JavaScript 在传统意义上被实现为一种解释语言，但在最近，它已经可以被即时编译（JIT）执行。随着最新的 HTML 5 和 CSS 3 语言标准的推行，它还可用于游戏、桌面、移动应用程序的开发和在服务器端网络环境运行，如通过 Node.js。

1.2.2 CSS 技术

层叠样式表（Cascading Style Sheets，简称 CSS），又称级联样式表、串接样式表。层叠样式表是一种用来为结构化文档（如 HTML 文档或 XML 应用）添加样式（字体、间距和颜色等）的计算机语言，由 W3C 定义和维护。目前最新版本是 CSS 3。

网页的用户和开发者都可以使用 CSS 来决定文字的颜色、字体、排版等显示特性。它使得网页更加生动，甚至可以使用不同的样式来控制网页上出现动画或其他效果。CSS 最主要的目的是将文件的内容与显示分隔开来，之后通过引入选择器的方式控制 DOM 的样式效果。这有许多好处：

- 文件的可读性加强。
- 文件的结构更加灵活。
- 用户和开发者可以自己决定不同的显示。
- 文件的结构简化了。

通过分离出来的 CSS 文件，可以针对每个不同的用户显示不同的效果，并且整个网站或其中一部分网页的显示信息被集中在一个地方，要改变它们很方便。针对不同的读者可以有不同的样式，比如有的用户需要的字体比较大，有的用户对于某些图标不可见或者是色弱等要对色彩做不同的调整。

CSS 还可以控制其他参数，例如声音（假如浏览器有阅读功能的话）或给视障者用的感受装置。

对于最新的 CSS 3，不但将 CSS 3 拆分为“模块”，而且在原有简单样式的基础上增加了一些新的属性，一些最重要 CSS 3 模块如下：

- 选择器。
- 盒模型。
- 背景和边框。
- 文字特效。

- 2D/3D 转换。
- 动画。
- 多列布局。
- 用户界面。

1.2.3 HTML（XML）技术

HTML（XML）技术其实是两种不同的技术，即 HTML 和 XML。

超文本标记语言（HyperText Markup Language，简称 HTML）是一种用于创建网页的标准标记语言。HTML 是一种基础技术，常与 CSS、JavaScript 一起被众多网站用于设计令人赏心悦目的网页、网页应用程序以及移动应用程序的用户界面。网页浏览器可以读取 HTML 文件，并将其渲染成可视化网页。HTML 描述了一个网站的结构语义，最新版本为 HTML 5。

可扩展标记语言（Extensible Markup Language，简称 XML），是一种标记语言。标记指计算机所能理解的信息符号，通过此种标记，计算机之间可以处理包含各种信息的文章等。如何定义这些标记，既可以选择国际通用的标记语言，比如 HTML，也可以使用像 XML 这样由相关人士自由决定的标记语言，这就是语言的可扩展性。XML 用到的主要有可扩展标记语言、可扩展样式语言（XSL）、XBRL 和 XPath 等。

XML 的诞生是为了取代 HTML，但由于 XML 一些烦琐的写法和不够灵活的方式，它并没有取代 HTML 的位置，反而因其清晰的结构和自定义的节点名称，被用于一些配置文件的书写和一种数据持久化的手段。

由于 HTML 和 XML 本源和初衷是一致的，所以对于两者的学习是通用的、可扩展的。

通过以上对 HTML 和 XML 的介绍，这里总结一下它们的联系与区别：XML 和 HTML 都用于操作数据或数据显示结构，在结构上大致相同，但在本质上的区别也相当明显。

1. 语法要求不同

（1）在 HTML 中不区分大小写，在 XML 中严格区分大小写。

（2）在 HTML 中，结构不严格，对于解析而言，如果能通过上下文清楚地显示出段落或者某一标签在何处结尾，那么可以省略</p>或者
之类的结束标签，甚至一些标签本身就是半闭合式的标签。但在 XML 中，必须遵循严格的树状结构，绝对不能省略结束标记。

（3）在 XML 中，拥有单个标记而没有匹配的结束标记的元素必须用一个“/”字符作为结尾。但 HTML 中这样的做法并不必要，反而在某些情况下增加“/”字符会出现不可预知的错误。

（4）在 XML 中，其属性值必须封装在引号中，所有的属性都必须带有相应的值，不能简写与

略写，而且有严格的书写规范。但在 HTML 中，引号是可用可不用的，并且有些属性名可以不包含任何的数值，这样的写法甚至可以自动生成相应的数值。

（5）在 XML 中，空白的部分认为是 XML 自身的一种内容，所以不会被解析器自动删除；但 HTML 中的空格是单个有效的，多余的空格将会被解析器过滤掉，成为无效标记。

2. 标记不同

（1）HTML 使用已经定义好的既有标记，但 XML 的语法结构是可以扩展的，所以并没有固有的标记。

（2）HTML 标签是预定义的，对于任意的标签都是存在含义和使用规范的，包括自定义的标签也应该提前解析；但 XML 的所有标签都是可以进行自定义、可扩展的，其属性也是扩展的一部分。

3. 作用不同

（1）HTML 是用来显示数据的；XML 是用来描述数据、存放数据的，所以可以作为持久化的介质。HTML 将数据和显示结合在一起，在页面中把数据显示出来；XML 则将数据和显示分开。XML 和 HTML 都是被设计用来描述数据，XML 的焦点是数据的内容，而 HTML 的焦点是数据的外观。

（2）XML 中的标签和属性是不具有任何的效果和操作的，更倾向于一种数据的表达形式和存储形式。而 HTML 更加倾向的是一种数据的显示形式和展示形式，所以这两种技术各有所长。

（3）通过 HTML（XML）与 CSS 文件的结合，可以构造出任意一种样式的网页和排版模式，比如对于微信小程序本身，就使用类似于 HTML（XML）的一套标签语言，所以通过微信小程序的 WXSS 和 WXML 的结合也可以构造出丰富的页面。

1.3 小程序开发所使用的技术

从 MVC 的传统开发模式来说，小程序本身其实只是对于 View 层的一种扩充，虽然它同样拥有一个逻辑层，但是由于小程序的体积以及应用方式的限制，所以逻辑层更多的是实现应用的显示方式和数据的整合，以及 API 的调用等操作，而非实现一个完整的应用。

小程序的开发并非是一种完全全新的开发形式和语言逻辑，对于任何一个拥有 HTML、CSS 和 JS 基础的开发者而言，小程序的开发是简单和便捷的，并没有学习的门槛和陡峭的学习曲线。如果读者并非一个网页开发者，也无须担心，因为就小程序本身的开发而言，是简单易用的。

注意：MVC 全名是 Model View Controller，是模型 Model、视图 View、控制器 Controller 的缩写，是一种软件设计典范。它用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑

聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

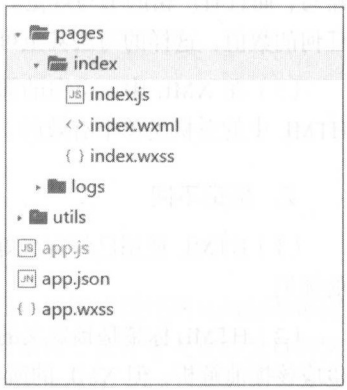


图 1-7　小程序开发项目目录

1.3.1　开发文件结构

对于微信小程序整个开发框架而言，其程序中包含一个描述整体程序的 App 和多个描述各自页面的 pages。小程序开发项目目录如图 1-7 所示。

图 1-7 是由微信小程序开发工具勾选 quickstart 选项后自动生成目录的一部分，也是小程序本身需要的一些文件组成。

一个小程序主体部分由三个文件构成（如表 1-1 所示），它们必不可少，必须放在项目的根目录上。

表 1-1　项目列表

文　件	必　填	作　用
app.js	是	小程序逻辑
app.json	是	小程序公共设置
app.wxss	否	小程序公共样式表

页面是开发者需要关注的地方，它存放在文件夹 pages 中，由于是系统自动生成的项目，其 index 文件夹中的文件就是一个常用页面的三个部分，而 log 文件夹是微信小程序自动生成的日志页面。

一个完整的小程序页面由四部分组成，具体的项目文件格式如表 1-2 所示。

表 1-2　项目文件格式

文件类型	必　填	作　用
js	是	页面逻辑
wxml	是	页面结构
wxss	否	页面样式表
json	否	页面配置

注意：对于微信小程序自身，一个页面的四个文件名称都是一致的，并且如果使用开发者工具，要在 app.js 中先注册这个页面，注册完成后会自动生成对应的三个同名文件（除去 json 文件）。

1.3.2　配置程序

app.json 文件是小程序中最基本的全局配置文件，它决定页面文件的路径（路由）、窗口表现、设置网络超时时间、设置多 tab 等。对于 app.json 的具体配置如表 1-3 所示。

表 1-3 app.json配置项

属 性	类 型	必 填	描 述
pages	String Array	是	设置页面路径
window	Object	否	设置默认页面的窗口表现
tabBar	Object	否	设置底部 tab 的表现
networkTimeout	Object	否	设置网络超时时间
debug	Boolean	否	设置是否开启 debug 模式

下面分别对每个配置项进行说明。

1. pages

pages 接受一个数组，每一项都是字符串，来指定小程序由哪些页面组成。每一项代表对应页面的（路径+文件名）信息，数组的第一项代表小程序的初始页面。小程序中新增/减少页面，都需要对 pages 数组进行修改。

文件名不需要写文件后缀，因为框架会自动去寻找路径.json、.js、.wxml、.wxss 的四个文件进行整合。

quickstart 项目的开发目录结构如图 1-7 所示，则需要在 app.json 中配置：

```
{
  "pages":[
    "pages/index/index"
    "pages/logs/logs"
  ]
}
```

2. window

window 配置用于设置小程序的状态栏、导航条、标题、窗口背景色，其具体包含的属性及配置如表 1-4 所示。

表 1-4 window属性及配置

属 性	类 型	默 认 值	描 述
navigationBarBackgroundColor	HexColor	#000000	导航栏背景颜色，如"#000000"
navigationBarTextStyle	String	white	导航栏标题颜色，仅支持 black/white
navigationBarTitleText	String	无	导航栏标题文字内容
backgroundColor	HexColor	#ffffff	窗口的背景色
backgroundTextStyle	String	dark	下拉背景字体、loading 图的样式，仅支持 dark/light
enablePullDownRefresh	Boolean	false	是否开启下拉刷新，详见页面相关事件处理函数

注意：HexColor（十六进制颜色值），如"#000000"。

比如对于第 5 章的实践示例，其 window 的配置如下：

```
{
  "window":{
    "backgroundTextStyle":"light",
    "navigationBarBackgroundColor": "#9999ff",
    "navigationBarTitleText": "Diary",
    "navigationBarTextStyle":"write"
  }
}
```

Windows 配置后的显示效果如图 1-8 所示。

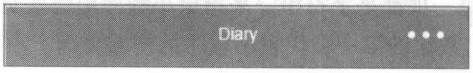


图 1-8　window 配置后的显示效果

3．tabBar

tabBar 用于配置小程序的菜单选项。如果小程序是一个多 tab 应用（客户端窗口的底部或顶部有 tab 栏可以切换页面），其本质相当于一个链接。可以通过 tabBar 配置项指定 tab 栏的表现，以及 tab 切换时显示的对应页面。

注意：通过页面跳转（wx.navigateTo）或者页面重定向（wx.redirectTo）所到达的页面，即使它是定义在 tabBar 配置中的页面，也不会显示底部的 tab 栏。

tabBar 是一个数组，只能配置最少 2 个、最多 5 个 tab，tab 按数组的顺序排序，其具体的配置属性如表 1-5 所示。

表 1-5　tabBar配置属性

属　性	类　型	必填	默认值	描　述
color	HexColor	是		tab 上的文字默认颜色
selectedColor	HexColor	是		tab 上的文字选中时的颜色
backgroundColor	HexColor	是		tab 的背景色
borderStyle	String	否	black	tabbar 上边框的颜色，仅支持 black/white
list	Array	是		tab 的列表，详见 list 属性说明，最少 2 个、最多 5 个 tab
position	String	否	bottom	可选值 bottom、top

tarBar 的属性 list 是一个数组，数组中的每个项都是一个对象，其属性值如表 1-6 所示。

表 1-6 list属性

属 性	类 型	必填	说 明
pagePath	String	是	页面路径，必须在 pages 中先定义
text	String	是	tab 上按钮的文字
iconPath	String	是	图片路径，icon 大小限制为 40KB，建议尺寸为 81px × 81px
selectedIconPath	String	是	选中时的图片路径，icon 大小限制为 40KB，建议尺寸为 81px × 81px

【示例 1-1】

如果对一个 quickstart 自动生成的项目增加如下配置，并且将其两个页面链接指向首页和日志页：

```
"tabBar": {
    "list": [{
      "pagePath": "pages/index/index",
      "text": "首页"
    }, {
      "pagePath": "pages/logs/logs",
      "text": "日志"
    }]
  }
```

其显示效果如图 1-9 所示。

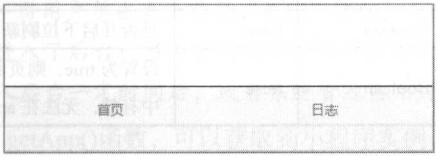


图 1-9 tarBar 显示效果

4. networkTimeout

networkTimeout 属性可以设置各种网络请求的超时时间，其具体属性如表 1-7 所示。

表 1-7 networkTimeout属性

属 性	类 型	必 填	说 明
request	Number	否	wx.request 的超时时间，单位毫秒，默认为：60000
connectSocket	Number	否	wx.connectSocket 的超时时间，单位毫秒，默认为：60000
uploadFile	Number	否	wx.uploadFile 的超时时间，单位毫秒，默认为：60000
downloadFile	Number	否	wx.downloadFile 的超时时间，单位毫秒，默认为：60000

5. debug

可以在开发者工具中开启 debug 模式，在开发者工具的控制台界面，调试信息以 info 的形式给出，其信息有 page 的注册、页面路由、数据更新、事件触发。目的是帮助开发者快速定位一些常见的问题。

6. page.json

每一个小程序页面也可以使用.json 文件来对本页面的窗口表现进行配置，这比 app.json 全局配置简单得多，只是设置 app.json 中的 window 配置项的内容，页面中配置项会覆盖 app.json 的 window 中相同的配置项。页面的.json 只能设置 window 相关的配置项，以决定本页面的窗口表现，所以无须写 window 这个键。

page.json 的配置属性如表 1-8 所示。

表 1-8　page.json的配置属性

属　性	类　型	默 认 值	描　述
navigationBarBackgroundColor	HexColor	#000000	导航栏背景颜色，如"#000000"
navigationBarTextStyle	String	white	导航栏标题颜色，仅支持 black/white
navigationBarTitleText	String	无	导航栏标题文字内容
backgroundColor	HexColor	#ffffff	窗口的背景色
backgroundTextStyle	String	dark	下拉背景字体、loading 图的样式，仅支持 dark/light
enablePullDownRefresh	Boolean	false	是否开启下拉刷新，详见页面相关事件处理函数
disableScroll	Boolean	false	设置为 true，则页面整体不能上下滚动；只在 page.json 中有效，无法在 app.json 中设置该项

1.3.3　逻辑层

对于微信小程序的逻辑层而言，其主要技术手段是 JavaScript，用于一些数据的定义和 API 方法的实现。

开发人员创建的任何一个页面，都至少有一个相关的 JS 文件，可以在文件中新建对此页面的操作和数据实现逻辑，该文件完全符合 JS 的语法和书写习惯，并且支持 ES 6，对于微信小程序运行的三大平台（iOS、Android 和用于调试的开发工具）的解析方式而言：

- 在 iOS 上，小程序的 JS 代码运行在 JavaScriptCore 中。
- 在 Android 上，小程序的 JS 代码通过 X5 内核来解析。
- 在开发工具上，小程序的 JS 代码运行在 nwjs（Chrome 内核）中。

虽然三大平台的环境十分相似，但至少在目前还是有一些区别的，这给开发者带来很大的困扰。

注意：在 0.10.101000 及之后版本的开发工具中，会默认使用 babel 将开发者代码 ES 6 语法转换为三大平台都能很好支持的 ES 5 代码，帮助开发者解决环境不同所带来的开发问题。开发者也可以在项目设置中关闭这个功能。ES 6(ECMAScript 6)是 JavaScript 最新标准，也称为 ECMAScript 2015。

1. 注册程序

App()函数用来注册一个小程序，它接受一个 object 参数，指定小程序的生命周期。object 参数说明如表 1-9 所示。

表 1-9 object参数说明

属 性	类型	描 述	触发时机
onLaunch	Function	生命周期函数——监听小程序初始化	当小程序初始化完成时，会触发 onLaunch（全局只触发一次）
onShow	Function	生命周期函数——监听小程序显示	当小程序启动，或从后台进入前台显示，会触发 onShow
onHide	Function	生命周期函数——监听小程序隐藏	当小程序从前台进入后台，会触发 onHide
onError	Function	错误监听函数	当小程序发生脚本错误，或者 API 调用失败时，会触发 onError 并带上错误信息
其他	Any		开发者可以添加任意的函数或数据到 object 参数中，用 this 可以访问

注意：前台、后台定义——当用户单击左上角的“关闭”按钮，或者按了设备 Home 键离开微信，小程序并没有直接销毁，而是进入了后台；当用户再次进入微信或再次打开小程序，小程序又会从后台进入前台，只有当小程序进入后台一定时间后，或者系统资源占用过高，小程序才会被销毁。

微信小程序提供了全局的 getApp()函数，可以获取到小程序实例，在定义了 App 的函数中，使用 this 即可获得实例。

注意：App()必须在 app.js 中注册，且不能注册多个。

2. 注册页面

Page()函数用来注册一个页面，同样接受一个 object 参数，其指定页面的初始数据、生命周期函数、事件处理函数等。

对于一个 page 拥有本身的函数和属性，具体的属性和说明如表 1-10 所示。

表 1-10 page属性和说明

属 性	类 型	描 述
data	Object	页面的初始数据

续表

属　性	类　型	描　述
onLoad	Function	生命周期函数——监听页面加载
onReady	Function	生命周期函数——监听页面初次渲染完成
onShow	Function	生命周期函数——监听页面显示
onHide	Function	生命周期函数——监听页面隐藏
onUnload	Function	生命周期函数——监听页面卸载
onPullDownRefresh	Function	页面相关事件处理函数——监听用户下拉动作
onReachBottom	Function	页面上拉触底事件的处理函数
onShareAppMessage	Function	用户单击右上角的“分享”按钮
其他	Any	开发者可以添加任意的函数或数据到 object 参数中，在页面的函数中用 this 可以访问

当定义一个页面之后，保存 app.json 文件，微信编辑器会自动生成 page 页面的代码，其代码如下：

```
//pages/index/home.js
Page({
  data:{},
  onLoad:function(options){
    //页面初始化，options 为页面跳转所带来的参数
  },
  onReady:function(){
    //页面渲染完成
  },
  onShow:function(){
    //页面显示
  },
  onHide:function(){
    //页面隐藏
  },
  onUnload:function(){
    //页面关闭
  }
})
```

上述代码自动生成的这些函数，我们称之为生命周期函数。

表 1-10 中的 onShareAppMessage 是用户分享函数，只有定义了此事件处理函数，右上角菜单才会显示“分享”按钮。用户单击“分享”按钮时会调用其方法，此事件需要返回一个 Object，用于自定义分享内容，可以自定义分享的字段如表 1-11 所示。

表 1-11 “分享”按钮字段

字 段	说 明	默 认 值
title	分享标题	当前小程序名称
path	分享路径	当前页面 path，必须是以“/”为开头的完整路径

【示例 1-2】

自定义分享的示例代码：

```
Page({
    onShareAppMessage: function () {
    return {
      title: '分享',
      path: '/page/index'
    }
  }
})
```

其显示效果如图 1-10 所示。

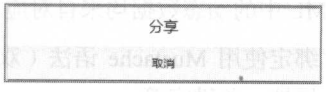


图 1-10 显示效果

3. 模块化——文件作用域

在 JS 文件中声明的变量和函数只在该文件中有效，不同的文件中可以声明相同名字的变量和函数，不会互相影响。通过全局函数 getApp()可以获取全局的应用实例，如果需要全局的数据可以在 App()中设置。

【示例 1-3】

设置在 globalData 的值：

```
globalData: 'I am global data'
```

上述定义的值在其他页面调用时，使用 getApp()获得全局 App 后调用：

```
var appInstance = getApp()
console.log(appInstance.globalData) //全局参数
```

模块化就是将一些公共的代码抽离成一个单独的 JS 文件，作为一个模块。模块只有通过 module.exports 或者 exports 才能对外暴露接口。

注意：小程序目前不支持直接引入 node_modules，开发者需要使用到 node_modules 时，建议复制相关的代码到小程序的目录中。

1.3.4 WXML

WXML 是类似于 HTML 的一套标签语言，结合基础组件、事件系统，可以构建出页面的结构。其语法和 HTML 如出一辙，通过既定的不同意义的标签作为页面的组成部分，并且通过解析器，解析出不同的展示样式。WXML 结合 WXSS 进行整个用户界面的布局和搭建。

【示例 1-4】

定义页面内容：

```
<text>Demo</text>
```

上述代码的效果如图 1-11 所示。这行代码符合 WXML 规范，意义是显示一行文字，而一个闭合<text>标签中所包含的内容，即为页面上显示的内容。

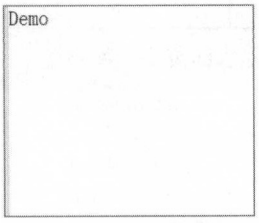


图 1-11 text 控件显示

WXML 可以完成数据绑定、运算、列表渲染、条件渲染、模板、事件、引用等操作。

1. 数据绑定

WXML 中的动态数据均来自对应 page 中的 data。

数据绑定使用 Mustache 语法（双大括号）将变量包起来，可以作用于内容、属性、关键字等。

【示例 1-5】

绑定一个 HelloWorld 字符串在页面中。

wxml 页面完整代码：

```
<view> {{ message }} </view>
```

JS 逻辑页面绑定的 data 属性，完整代码为：

```
Page({
  data: {
    message: 'Hello World!'
  }
})
```

本页的显示效果如图 1-12 所示。

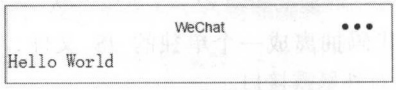


图 1-12 Hello World 的显示效果

注意：在 JS 中绑定关键字时，不要直接写 checked="false"，其计算结果是一个字符串，转成

boolean 类型后代表真值。

2. 运算

可以在{{}}内进行简单的运算，如算数运算、字符串运算、三元运算、逻辑判断、数据路径运算。

【示例 1-6】

在页面中绑定运算，wxml 页面完整代码：

```
<!--index.wxml-->
<view> {{a + b}} + {{c}} = 6 </view>
```

JS 逻辑页面绑定的 data 属性，完整代码为：

```
Page({
  data: {
    a: 1,
    b: 2,
    c: 3
  }
})
```

其显示效果如图 1-13 所示。



图 1-13 显示效果

3. 条件渲染

小程序中的条件语句是 wx:if。在框架中使用 wx:if="{{condition}}"来判断是否需要渲染该代码块，通过 JS 发送的数据条件实现前台模板的显示与否。其基本语法为：

```
<view wx:if="{{condition}}"> True </view>
```

也可以用 wx:elif 和 wx:else 来添加一个 else 块用于条件判定，具体使用和显示效果如下示例。

【示例 1-7】

小程序中的判断语句，wxml 代码：

```
<view wx:if="{{num == 1}}"> num 和 1 相等 </view>
<view wx:elif="{{num > 1}}"> num 大于 1 </view>
<view wx:else> num 小于 1 </view>
```

在 JS 文件中赋予 num 值为 2：

```
Page({
   data: {
    num:2
  }
})
```

其显示效果如图 1-14 所示。



图 1-14　显示效果

因为 wx:if 是一个控制属性，需要将它添加到一个标签上。但如果我们想一次性判断多个组件标签，可以使用<block/>标签将多个组件包装起来，并在上边使用 wx:if 控制属性。

【示例 1-8】

多个判断条件：

```
<block wx:if="{{true}}">
  <view> view1 </view>
  <view> view2 </view>
</block>
```

注意：<block/>并不是一个组件，它仅仅是一个包装元素，不会在页面中做任何渲染，只接受控制属性。

4．列表渲染

在组件上使用 wx:for 控制属性绑定一个数组，即可使数组中各项数据重复渲染该组件。数组当前项的下标变量名默认为 index，数组当前项的变量名默认为 item。

【示例 1-9】

使用 wx:for 的方式打印 9×9 乘法表：

```
<view wx:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" wx:for-item="i">
  <view wx:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" wx:for-item="j">
    <view wx:if="{{i <= j}}">
      {{i}} * {{j}} = {{i * j}}
    </view>
  </view>
</view>
```

其执行效果如图 1-15 所示。

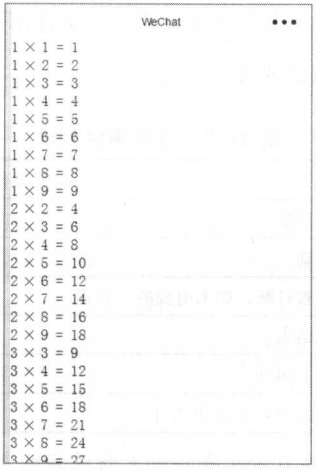


图 1-15 9×9 乘法表

类似 block wx:if，也可以将 wx:for 用在<block/>标签上，以渲染一个包含多节点的结构块。

【示例 1-10】

渲染多节点：

```
<block wx:for="{{[1, 2, 3]}}">
  <view> {{index}}: </view>
  <view> {{item}} </view>
</block>
```

5. 模板

WXML 提供了模板（template），可以在模板中定义代码片段，然后在不同的地方调用，达到一处页面代码多处使用的目的。

6. 事件

事件是视图层到逻辑层的通信方式，可以将用户的行为反馈到逻辑层进行处理。事件可以绑定在组件上。当事件被触发，就会执行逻辑层中对应的事件处理函数。事件对象可以携带额外信息，如 id、dataset、touches。

在组件中绑定一个事件处理函数 bindtap，当用户单击该组件的时候会在该页面对应的 page 中找到相应的事件处理函数。

事件分为冒泡事件和非冒泡事件。

- 冒泡事件：当一个组件上的事件被触发后，该事件会向父节点传递。

- 非冒泡事件：当一个组件上的事件被触发后，该事件不会向父节点传递。

WXML 的冒泡事件列表如表 1-12 所示。

表 1-12　冒泡事件列表

类　型	触发条件
touchstart	手指触摸动作开始
touchmove	手指触摸后移动
touchcancel	手指触摸动作被打断，如来电提醒、弹窗
touchend	手指触摸动作结束
tap	手指触摸后马上离开
longtap	手指触摸后，超过 350ms 再离开

注意：除表 1-12 之外的其他组件自定义事件如无特殊声明都是非冒泡事件，如<form/>的 submit 事件、<input/>的 input 事件、<scroll-view/>的 scroll 事件。

7. 引用

WXML 提供两种文件引用方式 import 和 include。

- import

import 可以在该文件中使用目标文件定义的 template。

【示例 1-11】

在 item.wxml 中定义了一个叫 item 的 template：

```
<!-- item.wxml -->
<template name="item">
  <text>{{text}}</text>
</template>
```

在 index.wxml 中引用了 item.wxml，就可以使用 item 模板。

```
<import src="item.wxml"/>
<template is="item" data="{{text: Hellow}}"/>
```

注意：import 有作用域的概念，即只会 import 目标文件中定义的 template，而不会 import 目标文件 import 的 template。

- include

include 可以将目标文件除了<template/>的整个代码引入，相当于是复制到 include 位置。

1.3.5 WXSS

WXSS 的文件主要就是对于 WXML 样式的一种补充内容，通过对于 WXML 中不同节点的不同样式描述，构建不同的页面样式。

【示例 1-12】

更改已经存在的 WXML 文件的内容，增加一个 class 属性，并且设定其值为“myText”：

```
<text class="myText">Demo</text>
```

在和其同名的 WXSS 文件中新增一个样式属性：

```
.myText{
      color:red
}
```

效果如图 1-16 所示。

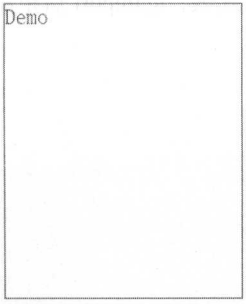


图 1-16 WXML 和 WXSS 结合

原本的 Demo 显示效果是黑色的 Demo 字样，通过增加 id 属性，在 WXSS 中获取到相关的节点，通过赋予样式 color：red，将整个文字颜色变为红色。

为了更适合开发微信小程序，微信小程序对 CSS 样式进行了扩充和修改，主要变动有：

- 尺寸单位。
- 样式导入。
- 内联样式。
- 选择器。
- 全局样式和局部样式。

1. 尺寸单位

尺寸单位是 rpx（responsive pixel），可以根据屏幕宽度进行自适应。规定屏幕宽为 750rpx。如在 iPhone 6 上，屏幕宽度为 375px，共有 750 个物理像素，则 750rpx = 375px = 750 物理像素，1rpx

= 0.5px = 1 物理像素。

常用设备和换算方式如表 1-13 所示。

表 1-13　常用设备和换算方式

设　备	rpx 换算 px（屏幕宽度/750）	px 换算 rpx （750/屏幕宽度）
iPhone 5	1rpx = 0.42px	1px = 2.34rpx
iPhone 6	1rpx = 0.5px	1px = 2rpx
iPhone 6 Plus	1rpx = 0.552px	1px = 1.81rpx

注意：在较小的屏幕上不可避免地会有一些毛刺，请在开发时尽量避免这种情况。因此，开发微信小程序时，设计师可以用 iPhone 6 作为视觉稿的标准。

2. 样式导入

使用@import 语句可以导入外联样式表，@import 后跟需要导入的外联样式表的相对路径，用;表示语句结束。

【示例 1-13】

导入外联样式表：

```
/** common.wxss **/
.small-p {
  padding:5px;
}
/** app.wxss **/
@import "common.wxss";
.middle-p {
  padding:15px;
}
```

3. 内联样式

框架组件上支持使用 style、class 属性来控制组件的样式。

静态的样式统一写到 class 中。style 接收动态的样式，在运行时会进行解析。

```
<view style="color:{{color}};" />
```

注意：请尽量避免将静态的样式写进 style 中，以免影响渲染速度。

4. 选择器

目前支持的选择器如表 1-14 所示。

表 1-14 支持的选择器

选 择 器	样 例	样例描述
.class	.classname	选择所有拥有 class="classname"的组件
#id	#idname	选择拥有 id=" idname "的组件
element	view	选择所有 view 组件
element, element	view, checkbox	选择所有文档的 view 组件和所有的 checkbox 组件
::after	view::after	在 view 组件后插入内容
::before	view::before	在 view 组件前插入内容

5. 全局样式与局部样式

定义在 app.wxss 中的样式为全局样式，作用于每一个页面。在 page 的 WXSS 文件中定义的样式为局部样式，只作用在对应的页面，并会覆盖 app.wxss 中相同的选择器。具体的 WXSS 样式请参考第 3 章。

1.4 搭建小程序开发环境

本节将会正式步入小程序的开发学习，讲解小程序开发环境的搭建，编写第一个小程序，并对代码进行详细解析。

1.4.1 IDE 下载

小程序最简单的开发方式是使用官方的小程序开发 IDE（Integrated Development Environment，集成开发环境），现在最新的版本为 0.11.112301。因为一直处于测试状态，所以 IDE 并非正式版本，但是可以方便读者的练习操作，并且自身集成调试功能和不同尺寸屏幕的测试，以及不同网络制式时的载入时间测试功能。

使用微信开发者工具可以方便实现的功能：

- 代码时样式自动补全选项默认开启，开发者可以主动关闭。
- 集成了 Chrome 的调试台环境，进行控制台的输出以及错误提示。
- 绑定开发者的微信号，方便调试以及将来的推送上线功能。
- 自动生成文件和 quickStart 方便开发。

微信小程序的开发工具一共有三个不同的版本，运行环境并没有依赖。只需要一个可以联网的环境，用于绑定开发者的微信，并且实现对于某一开发版本的推送功能。下面是具体的下载安装包的方法：

（1）打开任意的浏览器并输入地址（https://mp.weixin.qq.com/debug/wxadoc/dev/devtools/devtools.html），可以看到微信开发者工具文档，如图 1-17 所示。

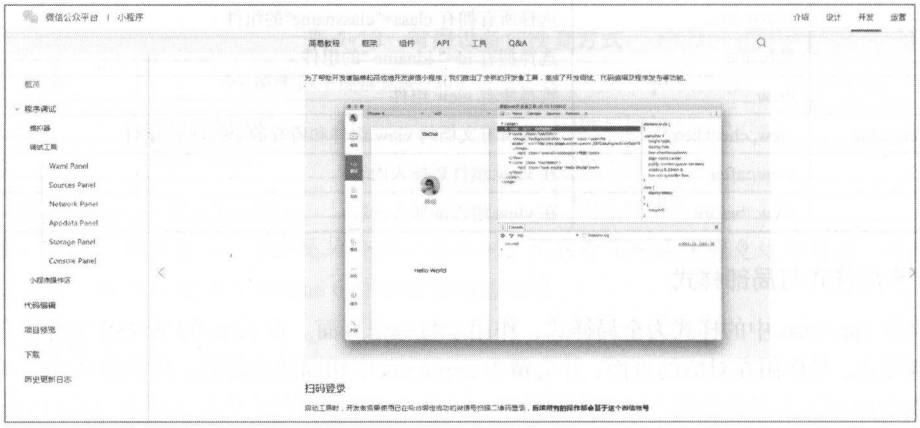


图 1-17　开发者工具文档

（2）单击左边菜单的“下载”选项，如图 1-18 所示。

（3）此时可以看到小程序的下载地址，选择适合的版本进行下载，如果使用 Windows 系统，可以右键单击“我的电脑”，在快捷菜单（如图 1-19 所示）中选择“属性”菜单来查看计算机的系统是 64 位还是 32 位。在“属性”窗口的右下方，可以看到整个电脑的系统配置，如图 1-20 所示。

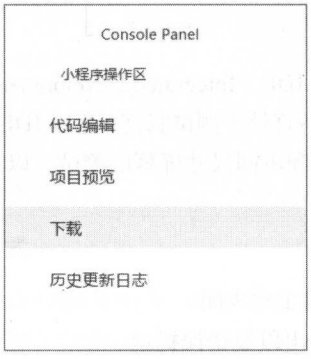


图 1-18　下载选项

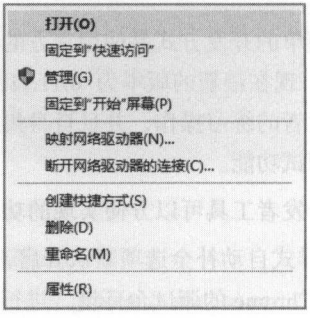


图 1-19　快捷菜单



图 1-20　系统配置

（4）单击适合自己的开发者版本后，浏览器会自动开始下载。如果浏览器没有自动下载，那么通过你跳转的页面网址全选复制，在迅雷或者是 IDM 等下载软件中新建下载任务即可，如图 1-21 所示。

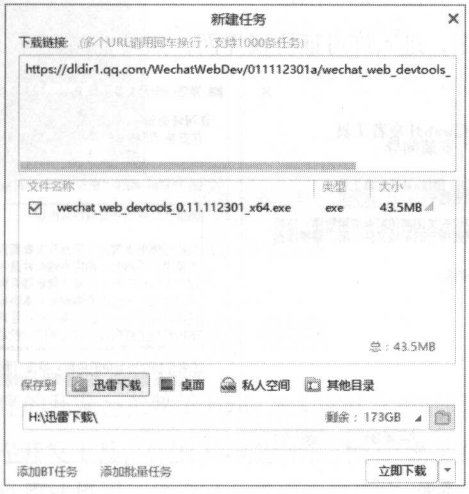


图 1-21 迅雷下载

（5）下载完成后即可打开下载到的目录，进行安装操作，具体步骤我们会在下一节以 Windows 为例进行说明。

1.4.2 Windows 系统的 IDE 安装

（1）打开下载的微信开发者工具目录，找到安装文件（如图 1-22 所示），双击打开。



图 1-22 小程序 IDE 安装包

（2）如果使用 Windows 10 或 Windows 8 的系统，可能会提示是否允许用户账户控制，选择“是”。

注意：如果安装不成功或者运行时出错，可能是用户权限不够导致的，可以在安装时右键单击安装程序，在菜单中选择“以管理员身份运行”选项，如图 1-23 所示。



图 1-23　右键菜单

（3）进入程序安装界面，如图 1-24 所示。

（4）单击“下一步”按钮，进入许可证协议界面，如图 1-25 所示。

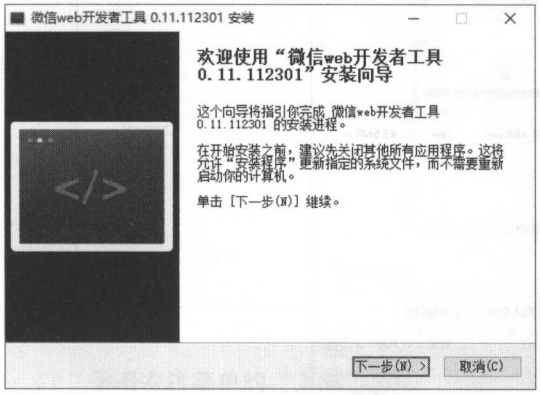


图 1-24　安装界面

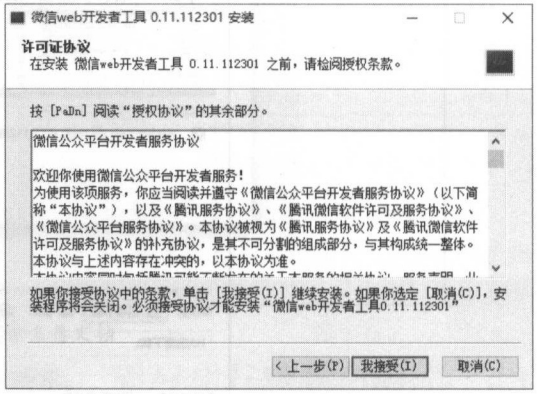


图 1-25　许可证协议界面

（5）接受协议才能继续安装，所以单击“我接受”按钮，打开图 1-26 的选定安装位置对话框。选择想安装的位置，单击“安装”按钮。

（6）正常安装组件时的界面如图 1-27 所示。

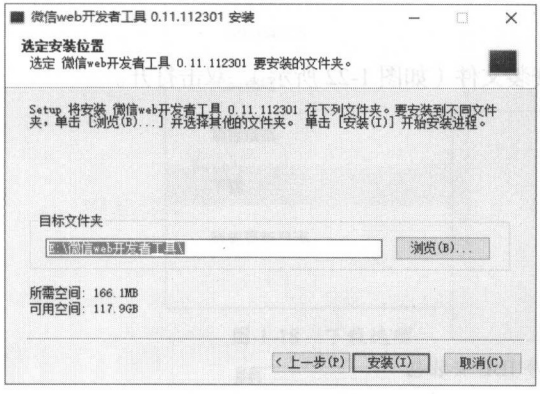


图 1-26　安装界面

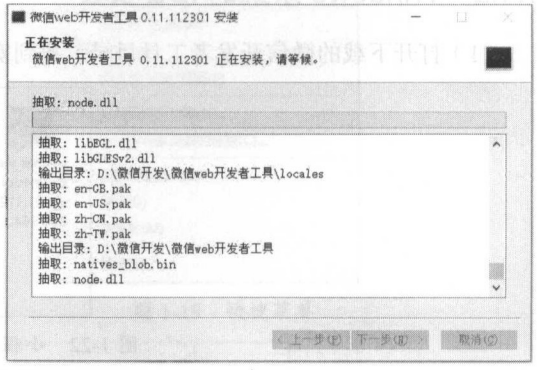


图 1-27　安装界面

（7）等待片刻后，安装完成，如图 1-28 所示。

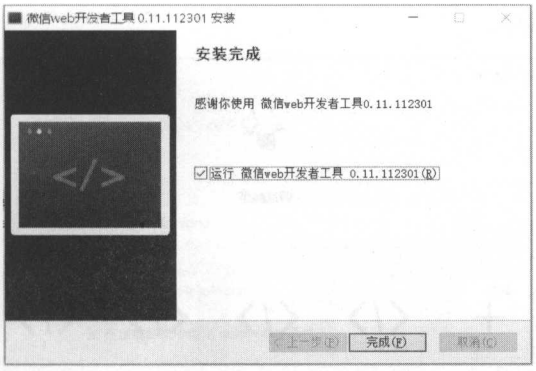


图 1-28 安装完成

1.5 上手实践：第一个 Hello WX 程序

本节正式进入微信小程序的开发，主要是 Hello WX 的开发和基础代码的解析学习。

（1）双击打开微信小程序开发工具，如图 1-29 所示，这个界面是微信小程序的初始化界面。因为微信小程序是一种强联网性质的应用，并且完全集成在微信中，所以微信小程序开发必须登录开发者的微信账号，这样也可以更加方便进行未来的应用发布和更新。



图 1-29 初始化界面

（2）使用自己的手机打开微信，扫描初始页二维码。单击确认绑定登录，此时打开微信开发者工具就可以开发了。如果之前没有进行过开发工作，打开时没有任何的项目。每一次新建的项目都

会出现在图 1-30 这个界面中。

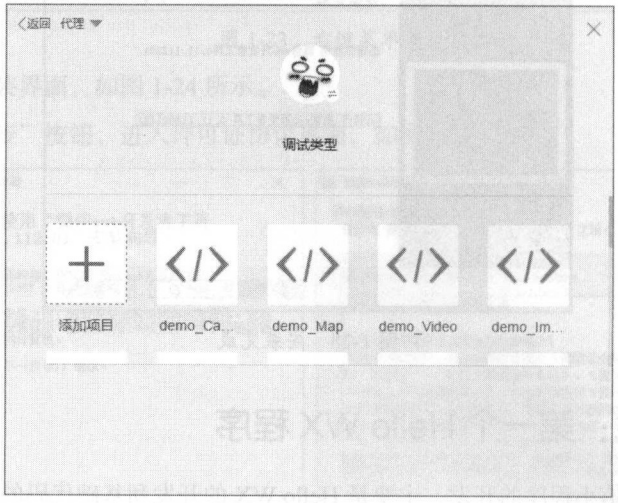


图 1-30　项目目录

（3）单击“添加项目”按钮，出现新建项目页面，如图 1-31 所示。在 AppID 处选择无 AppID，AppID 的作用就是某个应用唯一的 ID 号码，可以用在将来的上线审核和更新等的应用信息识别。对于应用的开发，除了一些特殊的 API 和上线以外，对于开发并没有太大的影响，这里选择无。填写相关的项目名称和开发的项目地址，并且勾选“在当前目录中创建 quick start 项目”。填写完成后，效果如图 1-32 所示。

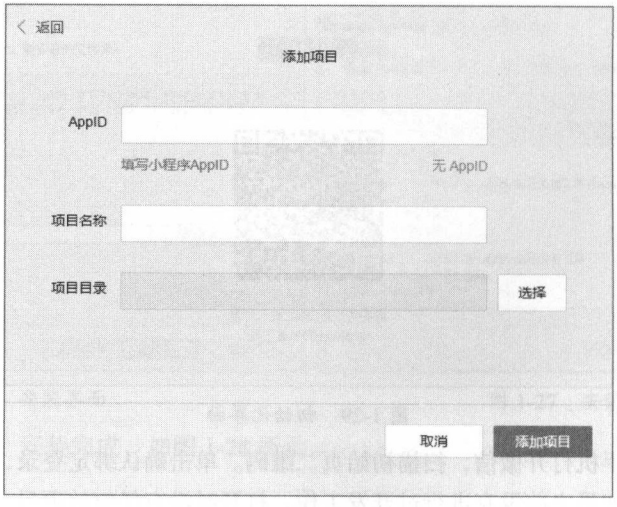


图 1-31　新建项目

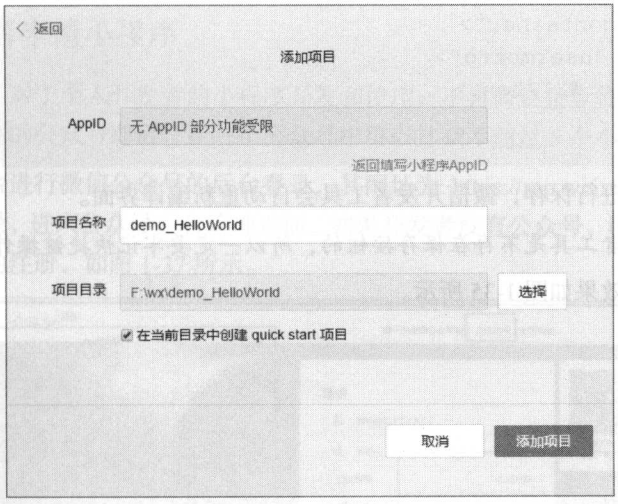


图 1-32 项目实例

（4）单击“添加项目”按钮进入开发界面，如图 1-33 所示。此时并不是一个空应用，而是系统自动生成的一个带了必需文件的应用。

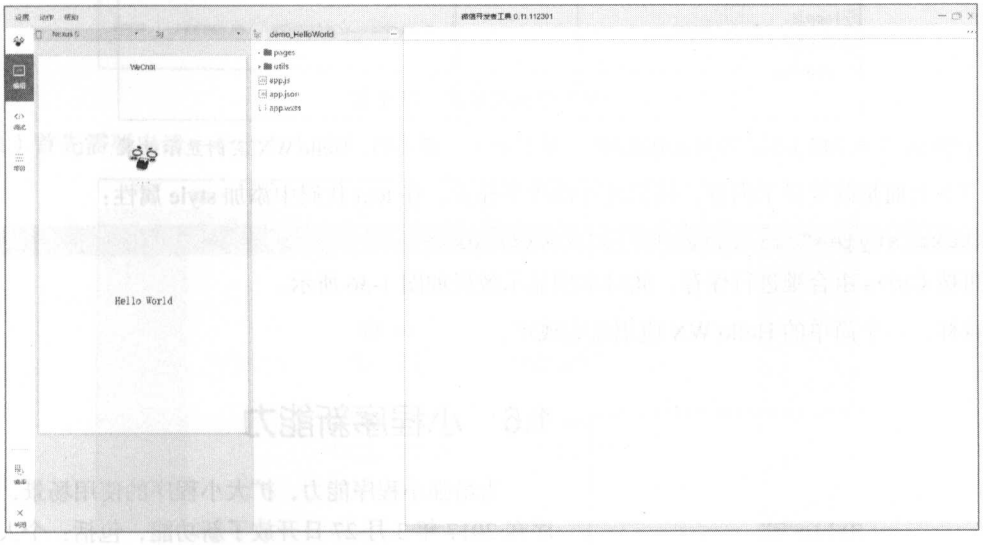


图 1-33 开发界面

（5）在开发界面上选择 pages/index 文件夹下的 index.wxml 文件，如图 1-34 所示，此时会在右边自动打开代码编辑器。可以看到 index.wxml 中的代码，可以对其进行编辑。

（6）尝试将代码更改为：

```
<view class="container">
  <view class="usermotto">
    <text>Hello WX</text>
  </view>
</view>
```

按 Ctrl+s 组合键进行保存，微信开发者工具会自动重新编译界面。

注意：微信开发者工具是不存在保存按钮的，所以一定要牢记快捷键操作。

此时，本例显示效果如图 1-35 所示。

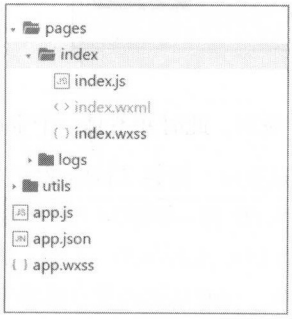


图 1-34　项目文件结构

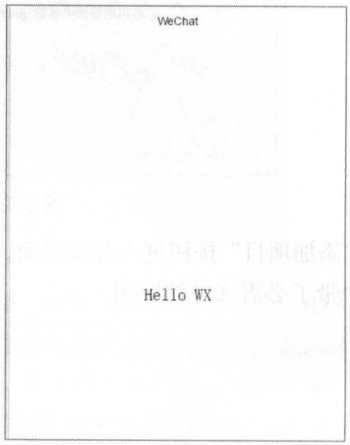


图 1-35　Hello WX 实例显示效果

（7）上面是改变显示内容，我们还可以改变样式，在 text 代码中添加 style 属性：

```
<text style="color:red">Hello WX</text>
```

再按 Ctrl+s 组合键进行保存，此时本例显示效果如图 1-36 所示。

这样，一个简单的 Hello WX 应用就完成了。

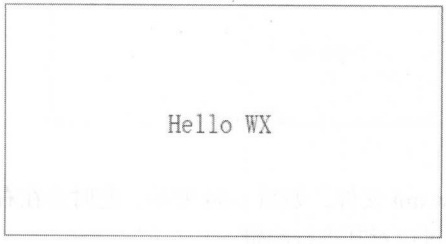


图 1-36　结合颜色的实例

1.6　小程序新能力

为增强小程序能力，扩大小程序的使用场景，小程序在 2017 年 3 月 27 日开放了新功能，包括：个人开发者可申请小程序，使用公众号中的一个链接便可以打开小程序本身等。这让小程序对于公众号的支持和联系变得更加紧密，不再需要通过专用的微信小程序入口进入相关小程序。本节就来介绍这些新能力。

1.6.1 个人开发者申请小程序

微信小程序开放了对于个人开发者的小程序开发和使用，不再需要企业资质和相关的标准，只要用户拥有一个实名制的公众号即可，在一个公众号中可以挂载不超过 5 个小程序。

进入微信公众平台进行微信公众号的后台登录，其网址为 https://mp.weixin.qq.com/，输入相关的用户名和密码并登录，进入公众号后台管理页面，如果开发者没有公众号，可以根据下面的步骤，进行小程序和公众号的注册，如图 1-37 所示。



图 1-37 微信公众号注册

（1）首先需要申请一个个人小程序，单击“立即注册”按钮进入注册页面，如图 1-38 所示。

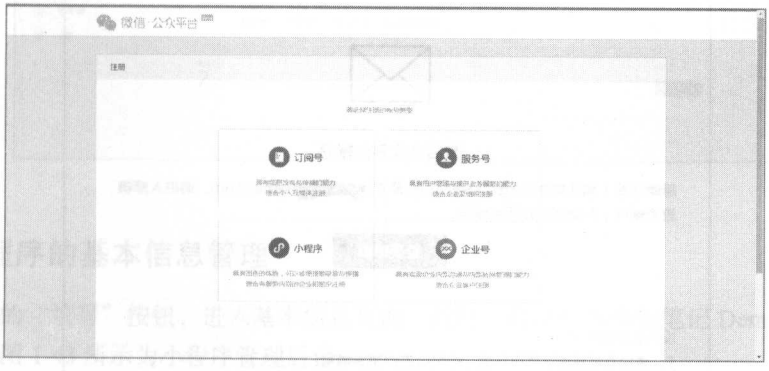


图 1-38 注册页面

（2）选择“小程序”选项，进入小程序的资料填写页面，如图 1-39 所示。输入与原有微信公众号相同的身份信息后，选中“协议”复选框，然后单击“注册”按钮。

注意：如果申请者已经拥有公众号的账号，那么此小程序的邮箱不能和公众号的邮箱是同一个。

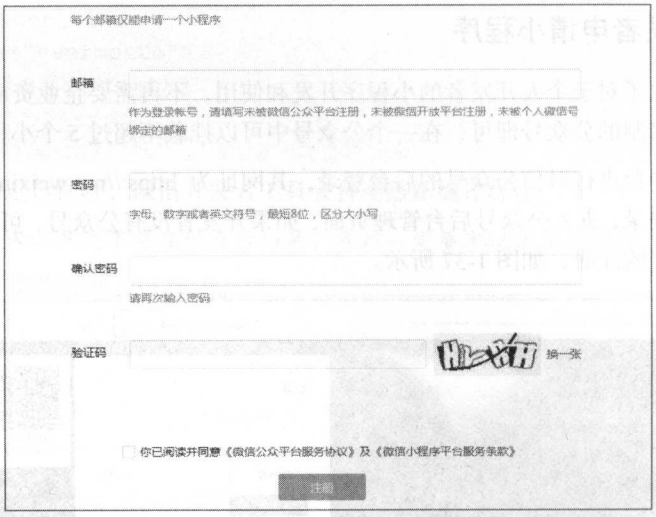


图 1-39　申请小程序页面

（3）注册成功后，会收到一封激活邮件，如图 1-40 所示。单击邮件中的链接，激活小程序，如图 1-41 所示。



图 1-40　邮箱激活

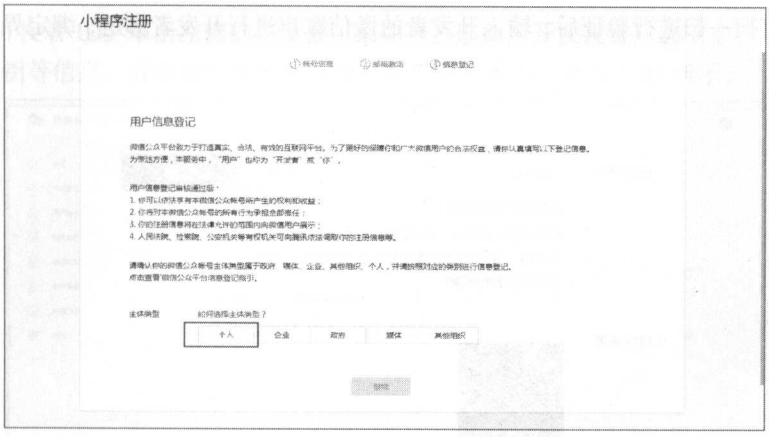


图 1-41 激活小程序

（4）此时需要选择账号的主体，这里选择个人，个人主体会要求用户输入正确的身份证号、姓名、手机号等内容，并且需要绑定一个管理员微信。单击“继续”按钮，提交注册信息，进入小程序管理后台，如图 1-42 所示。



图 1-42 小程序管理后台

1.6.2 小程序的基本信息管理

单击主页的“填写”按钮，进入基本信息页面，此时，可以将本书的笔记 Demo 作为第一个小程序上传，如图 1-43 所示为小程序管理后台。

注意：小程序不允许重名。

针对上传的小程序，开发者可以进行如下设置。

（1）绑定相关的小程序开发者。对于小程序而言，一个小程序本身允许绑定 10 个账户作为开

发者。通过微信扫一扫进行验证后，输入开发者的微信账户进行开发者绑定，绑定界面如图 1-44 所示。



图 1-43　小程序管理后台

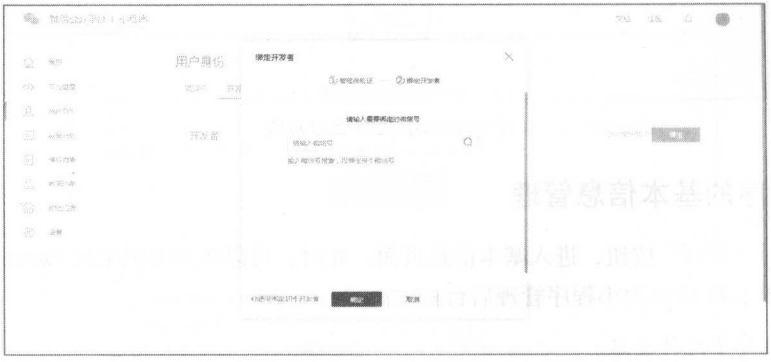


图 1-44　绑定开发者的界面

注意：对于微信的管理员而言，自动拥有开发者和体验者权限。

（2）查看开发者 ID。单击左侧的“设置”菜单项，选择“开发设置”选项卡，可以看到小程序的 AppID 和密钥等信息，并且能在此页面中添加服务器的域名，如图 1-45 所示。

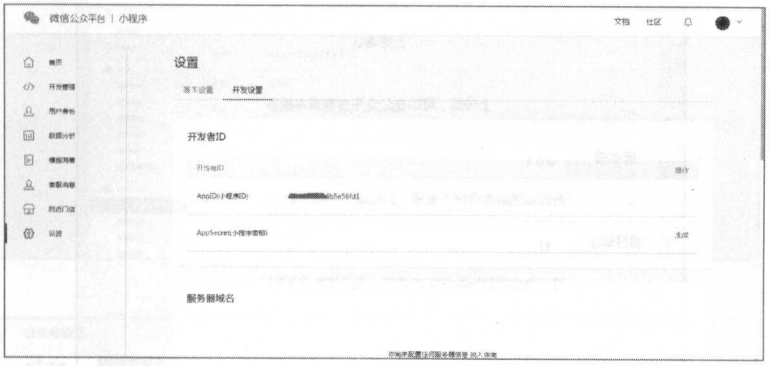


图 1-45 AppID 获取

获取 AppID 后，在开发工具中新建小程序项目时，通过绑定相关的 AppID（如图 1-46 所示），即可进行微信小程序的开发，使用上传发布等功能。

（3）在开发者工具中还包含代码的上传、已经使用者的预览等功能，如图 1-47 所示。

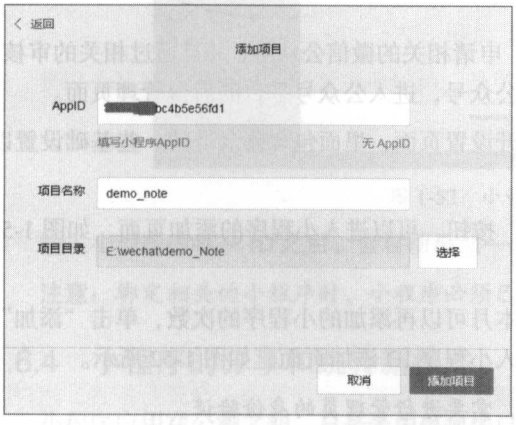


图 1-46 新建小程序



图 1-47 预览小程序

单击“上传”按钮，在验证用户身份后，输入版本号和相关的备注信息，即可上传相关的小程序，如图 1-48 所示。

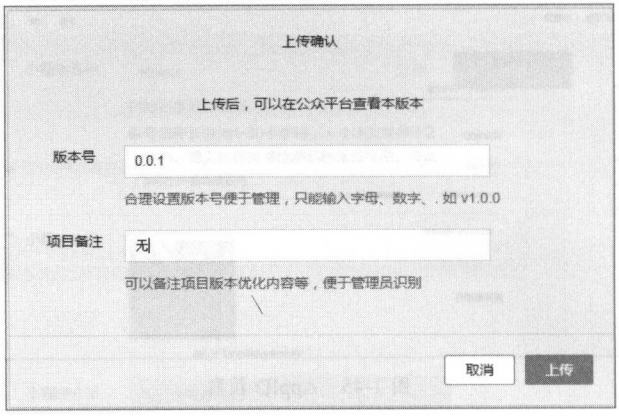


图 1-48　上传小程序

同样，在开发版本中会显示出此版本的信息，如图 1-49 所示。



图 1-49　开发版本

1.6.3　已有公众号绑定小程序

如果没有公众号，首先登录微信的公众号平台，申请相关的微信公众号，并且通过相关的审核，此验证信息需要和小程序的信息相同。如果已经有公众号，进入公众号平台的后台管理页面。

（1）单击“设置”下的“公众号设置”菜单打开设置页面，里面包含公众号的一些基础设置以及小程序的添加，如图 1-50 所示。

（2）单击“相关小程序选择”条目后的“添加”按钮，可以进入小程序的添加页面，如图 1-51 所示。

（3）此页面会显示出已经添加的小程序，还有本月可以再添加的小程序的次数，单击“添加”按钮，可以添加一个新的小程序。验证身份后会进入小程序 ID 添加页面，如图 1-52 所示。

注意：此操作对于微信公众号是一次敏感操作，需要进行管理员的身份验证。



图 1-50 设置界面

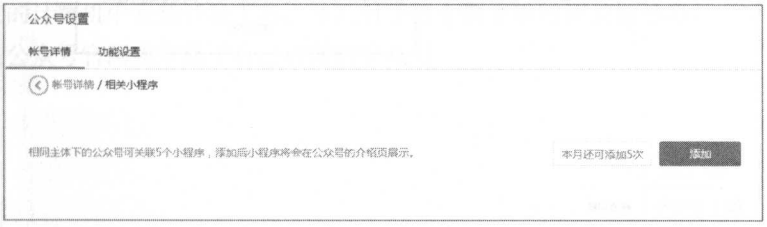


图 1-51 小程序添加



图 1-52 小程序 ID 添加页面

输入小程序的唯一 ID 之后，会自动添加一个小程序。

注意：绑定相关的小程序时，小程序必须已经处于发布状态。

1.6.4 小程序的分享和跳转新模式

小程序应用在公测之初，只是采用单独接口的方式进行小程序的分享和使用，用户必须通过分享的小程序专用链接或者小程序入口的方式进入小程序。而如今（截至 2017 年 4 月），小程序增加

了新的分享模式。

1. 公众号自定义菜单可打开相关小程序

已认证的公众号，可将已关联的小程序页面放置到自定义菜单中，用户单击后可打开该小程序页面。公众号运营者可在公众平台进行设置，也可以通过自定义菜单接口进行设置，如图 1-53 所示。

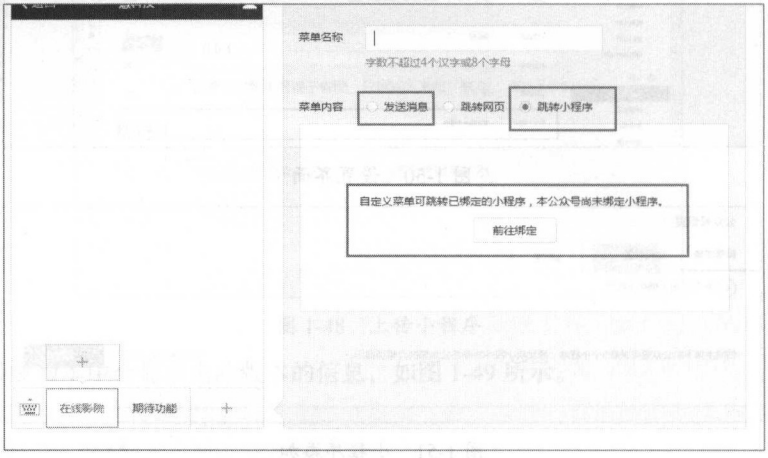


图 1-53　自定义菜单

2. 公众号模板消息可打开相关小程序

公众号已关联的小程序页面可以配置到公众号的模板消息中，用户单击公众号下发的模板消息，可以打开对应的小程序页面。

3. 公众号关联小程序时，可选择给粉丝下发通知

公众号关联小程序时，可选择给粉丝下发通知消息，粉丝单击该通知消息可以打开小程序。该消息不占用原有群发条数。

4. 移动 App 可分享小程序页面

开发者可以把小程序绑定到微信开放平台。绑定后，同一微信开放平台账号下的 App 可分享已绑定的小程序页面到微信内的会话或群聊。

5. 扫描普通链接二维码可打开小程序

商户如果在线下已铺设了普通链接二维码，可在公众平台的小程序管理后台进行配置，用户扫描该原有线下普通链接二维码，可直接打开小程序。

1.7 小结

本章通过对小程序相关的技术进行介绍和学习，使读者明白小程序本身的技术手段和开发模式，因为小程序本身涉及的技术是非常广泛的，本书只从入门部分进行一些粗浅讲解，具体的技术深挖，还需要读者多方面学习和练习。

本章完成开发者工具的下载和安装，并且通过入门的 Hello WX 例子，解析出小程序的开发步骤和开发流程，读者有了这些基础可以更好地进行后面的学习。

1.8 实践与练习

- 在自己的计算机中安装开发工具，并且对于每个例子进行相关的练习。
- 在微信公众号官网建立属于自己的微信公众号。

2 熟悉小程序开发工具

本章学习的是微信小程序开发工具和环境的详细介绍，通过本章的学习，读者能够充分地理解并熟练地使用相关的开发工具进行微信小程序的开发，开发中也可以进行调试并理解控制台的错误信息。

本章涉及的主要知识点有：

- 微信小程序开发工具的结构。
- 微信小程序开发工具的调试功能。
- 微信小程序开发工具的错误提示分析和基本操作。
- 微信小程序项目的创建和管理。

2.1 熟悉微信开发者工具

本节主要介绍开发者工具中各种窗口的作用和功能，目的是让读者可以通过调试窗口进行小程序的调试工作和基础的错误修正。

2.1.1 微信开发 IDE 简介

对于任何程序的开发而言，都需要一个与之相对的 IDE，IDE 的使用可以极大提高编程效率和编程开发速度，减少很多不必要的麻烦。

对于微信小程序的开发，官方也出品了对应的 IDE，PC 端基于 Chrome 内核的开发者工具，不仅仅提供了一个用于用户书写代码的环境，更是在其中增加了调试、代码高亮、项目管理、代码提示、自动完成等功能。

对于微信小程序开发工具而言，由于部分 API 是通过模拟返回的，所以并不一定能显示出真实

的用户信息或者需要的数据。但绝大部分的 API 均能在模拟器上呈现出正确的状态。

微信开发者工具主要有三大模块，分别是编辑模块、调试模块、项目模块，位于开发者工具的左侧，呈菜单状，如图 2-1 所示。只要单击不同的菜单，就能切换右边不同的工作视图，方便完成代码的编写和功能的调试，并且处于开发视图的按钮将会以绿色的高亮进行区分和显示。



图 2-1 微信开发者工具的三大模块

注意：微信小程序现阶段处于一个高速迭代的时期，所以对于开发者工具而言，其功能和特性也处于一个高速迭代的时间点。

2.1.2 IDE 调试窗口

既然是 IDE，那么集成一个方便调试的功能是最必要的功能之一。对于这一点，微信开发者环境是大大优于其他第三方工具的。单击左侧的调试菜单可以切换右侧的工作视图进入调试界面，同时，调试按钮将会以高亮的方式显示，如图 2-2 所示。整个界面分为三个部分，左边的小程序展示部分是小程序 IDE 中编辑界面和调试界面的公共部分，其负责实时展示小程序的运行情况，方便开发者进行相应的调试。

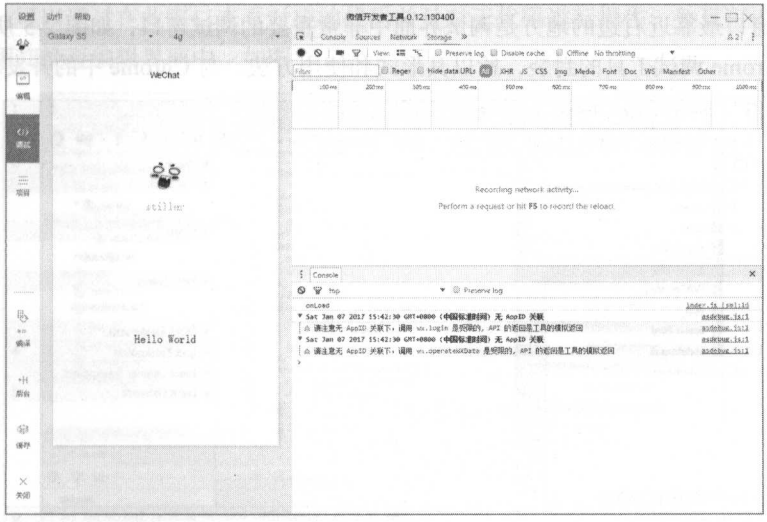


图 2-2 调试界面

其运行效果会因为用户的保存或是重启进行更新，当鼠标移动到此页面中，会变化成一个圆形的阴影状用于模拟用户触摸操作。

对于不同设备、不同屏幕大小的调试，微信小程序开发工具提供了不同的选项方便开发者进行调试，其菜单位于运行界面的上端，如图 2-3 所示。开发对应的小程序时，可以根据不同的运行设

备需求进行选择。

为了方便掌握测试页面在不同网络环境中的打开情况，在设备选择菜单的旁边，增加了多网络环境的选择，可以通过对于网络环境的选择，查看小程序自身在不同网络环境中的体验效果和加载速度，如图 2-4 所示。

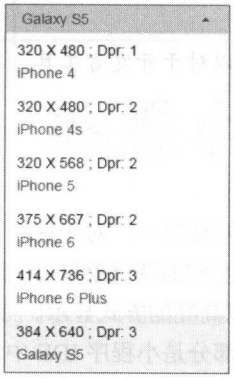


图 2-3　设备选择

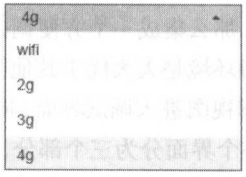


图 2-4　网络环境选择

整个调试窗口最靠近右边的地方是调试界面中非常重要的调试窗口，如图 2-5 所示。这个调试窗口相当于 Chrome 调试工具的封装，所以其形式和使用方法，与 Chrome 中的开发者工具相同。

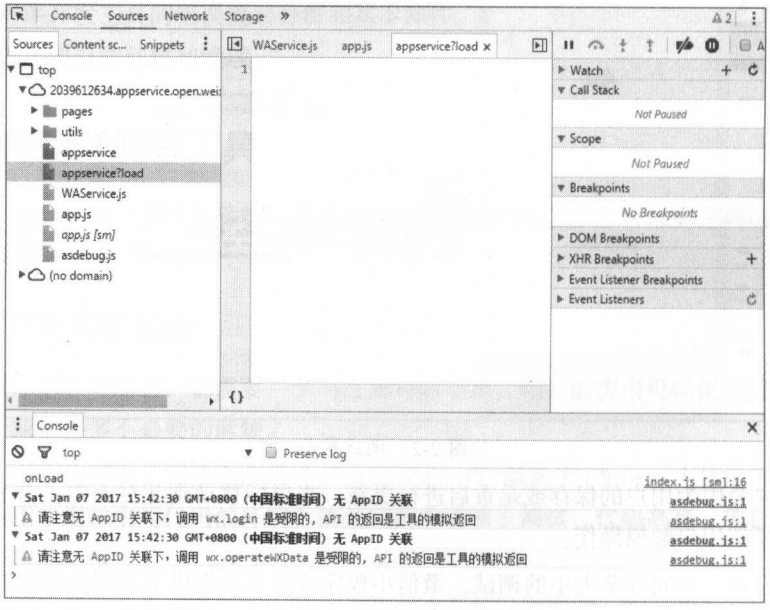


图 2-5　调试窗口

这里介绍调试窗口常用的一些功能。

（1）首先是 Console 选项卡，其主要是对于 JS 的控制台和输出错误和警告信息。在小程序代码中使用 console.log()方法可以在此控制台窗口中打印输出的值和调试信息，然开发者可以进行程序的调试工作。具体显示效果，如图 2-6 所示。

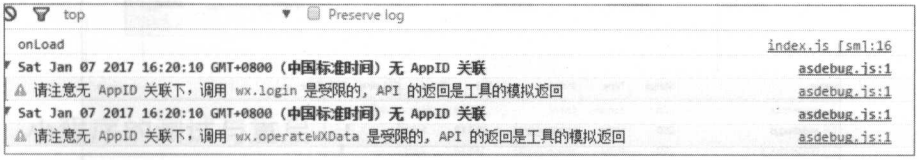


图 2-6 控制台

注意：控制台中的显示效果，黄色为程序警告，红色为程序错误，黑色字体为打印信息和详情信息。

为了方便开发者在不同界面中的调试和输出测试，Console 选项卡会在用户选择其他调试选项卡时，自动更换位置在整个调试窗口的右下方处，如图 2-5 所示。

（2）Console 选项卡旁边是 Sources 选项卡，顾名思义，单击它会出现整个项目的代码和对于代码的调试选项，可以通过在此窗口查看对应的代码并且对于代码进行操作，并且可以打开并查看相关的代码而不用切换至编辑界面中，如图 2-7 所示。

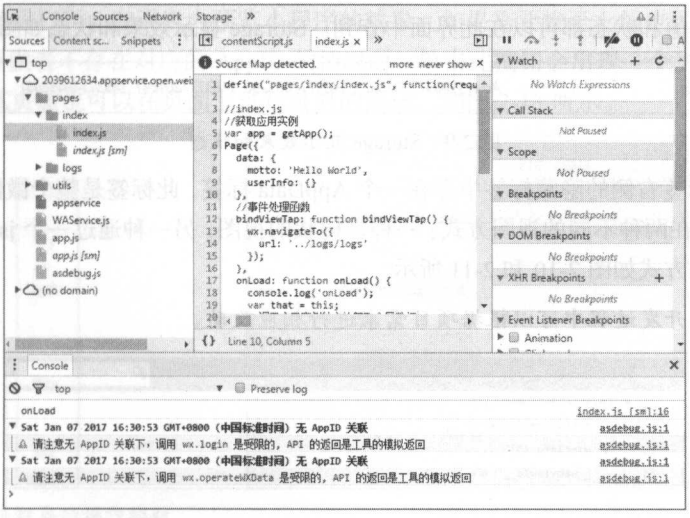


图 2-7 在调试窗口查看代码

（3）Network 选项卡也是非常重要的一个调试功能，其主要是用于小程序本身对于网络、自身页面和资源请求的一个记录的调试功能，界面中提供了详细的请求格式、返回码、文件大小及类型

等数据信息。对于时间的请求以图的形式表现出来，通过对这些数值的分析，可以有效且快速地发现并定位到程序的性能瓶颈，以及请求不通的错误页面。Network 选项卡的界面如图 2-8 所示。

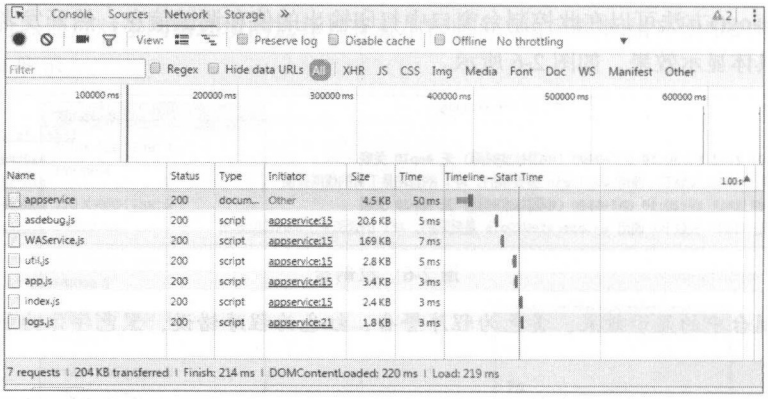


图 2-8　Network 选项卡的界面

注意：所有页面请求的信息都会出现在此选项卡中，鼠标悬停至请求资源的条目上方，会自动提示请求的地址信息。

（4）Storage 选项卡是对于整个小程序的一个临时存储数据的视图，这里不同于 Chrome 中的 cookie 和 session 等存储方式，微信小程序对于自身的存储方式进行了封装，不再区分存储。所有在程序中临时存储的值和状态都可以在此界面中查到。Storage 显示效果和状态如图 2-9 所示。

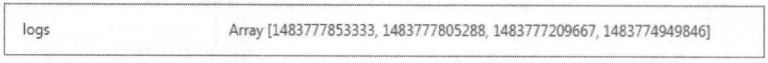


图 2-9　Storage 显示效果和状态

（5）Storage 标签右侧的隐藏标签中存在一个 AppData 标签，此标签是整个微信小程序的一些简单配置信息，它存在两种不同的视图方式：一种是树状的视图；另一种通过一个 json 配置进行显示。此标签的两种视图方式如图 2-10 和 2-11 所示。

注意：在实际开发过程中可以根据项目需求进行相应改动。

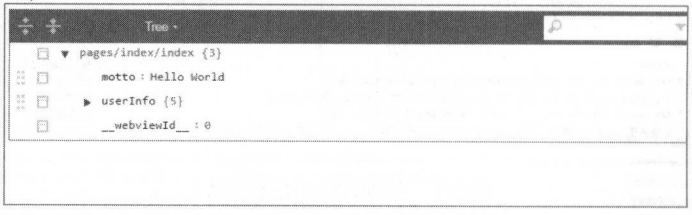


图 2-10　树状视图

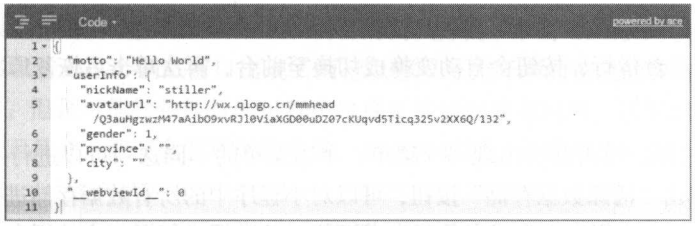


图 2-11 代码视图

2.1.3 小程序的测试与重启

所有的小程序测试都是在小程序的开发工具中完成的。对于任意的小程序，其实是遵守规范的网页类的应用模式，所以除去一些虚拟测试环境中没有的传感器等功能以外，其余的测试均可在开发工具中实现。

小程序的测试主要使用开发工具中左侧下方的四个按钮，通过对于不同测试环境的更改来实现对小程序的测试。具体的按钮如图 2-12 所示。

（1）未命名的“重启”按钮，通过单击这个按钮，当前小程序将会重新载入，相当于删除所有的缓存重新对该程序进行开启测试。

（2）“编译”按钮，单击此按钮，如果已经关联相关的 AppID，可以生成一个预览效果页面，开发者可用本人微信扫码，在手机内预览小程序的效果。如果需要推至上线也可以使用此按钮进行版本的提交。当开发者不存在对应的 AppID 的情况时，单击此按钮会出现一个测试页面的输入，通过输入相应的测试页面也可以在页面上进行真机的测试，如图 2-13 所示。



图 2-12 小程序的操作按钮

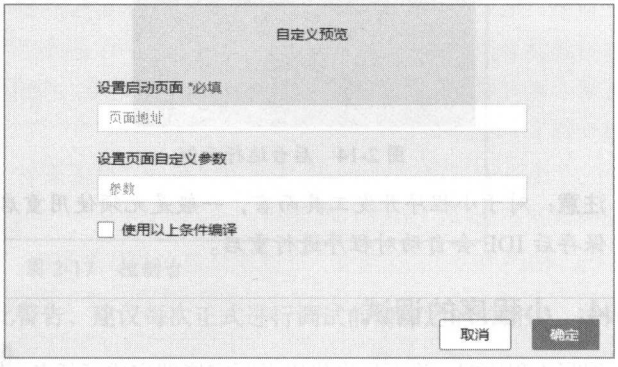


图 2-13 输入测试页面

（3）“后台”按钮，单击此按钮可以模拟手机中小程序被挂起时的状态，即在手机的后台时小程序的运行情况，包括对小程序后台功能是否休眠，是否重新进入活跃数据，是否继续存在等情况

进行对应的调试和修改。单击“后台”按钮后整个调试的显示页面将会覆盖上一层黑色半透明蒙版，并且提示程序正在后台运行，按钮会自动变换成切换至前台，再次单击后恢复原状。具体效果如图 2-14 所示。

（4）“缓存”按钮，单击其会出现一个菜单，单击菜单的不同选项可以进行不同的缓存删除，如图 2-15 所示。单击“清除数据存储”按钮，可以对小程序中的所有数据存储进行清除，因为包括从 API 中获得的数据，小程序本身或者是开发者可能对此数据进行了相应的缓存，可能取不到更新后的数据。用此方法可以将所有的数据缓存清除，获得更新后的数据信息。

同样，文件可能出现缓存及权限授权信息的缓存，正确清理相关的缓存，可以清除小程序开发过程中的一些缓存导致的错误。

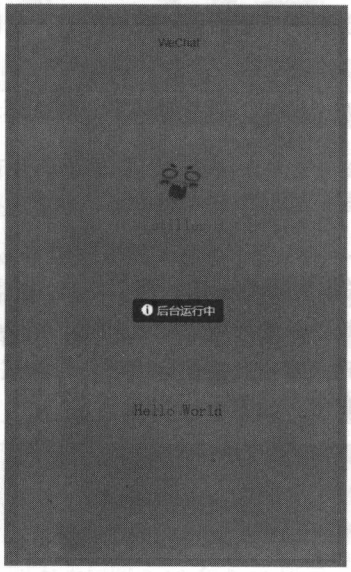


图 2-14　后台运行示例

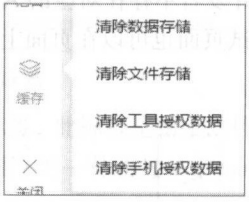


图 2-15　缓存按钮

注意：对于小程序开发工具而言，一般是无须使用重启按钮对整个项目进行重启，在编辑代码时，保存后 IDE 会自动对程序进行重启。

2.1.4　小程序的调试

编写小程序时，并不是将代码书写完毕即完成了任务，其中的调试和测试部分的重要性和时长，远远超过了书写代码的部分。前面已经介绍了相关的调试功能和界面，本节将会介绍相关的调试应用。

（1）如图 2-16 所示，先建立一个项目。使用微信小程序开发工具自带的快速开始进行测试演示。

（2）单击“调试”按钮，进入 Hello World 项目的调试界面。单击 Console 选项卡，进入控制台，如图 2-17 所示，可以看到这个调试窗口出现了两行以感叹号为开端的黄色的警告，警告是不会影响程序的正确执行的，但是可能导致执行的过程中出现某些错误或者问题。这里的警告主要是提示开发者，微信开发者工具并不能完全实现微信中的一些 API 返回，是模拟返回的。

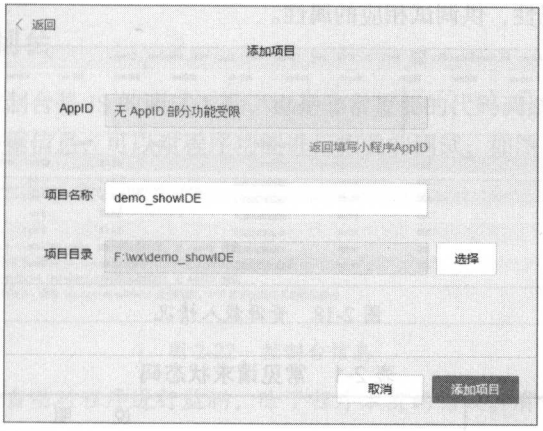


图 2-16 新建项目

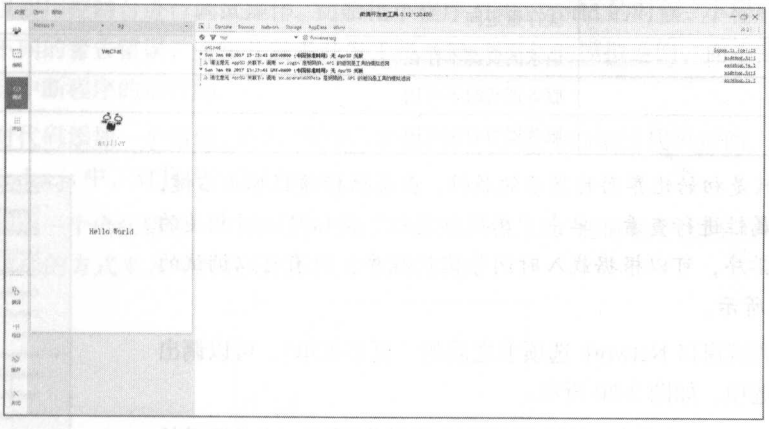


图 2-17 控制台

（3）单击“清除”按钮，可以清除此警告，建议每次正式进行调试前清除之前的错误，这样不会导致调试时历史错误和此次调试的混淆。

（4）单击 Network 选项卡，可以观察小程序的载入情况，选项卡中的红色小圆点是对于网络监听的启用，默认是选中状态，再次单击时会取消选中。取消选中将不会记录此小程序页面以及资源的获取情况。

在 Network 记录的情况下，按 F5 键刷新页面，可以获取此小程序资源的载入情况，如图 2-18 所示。这里主要是对于资源的获取状态的一个展示，其 Name 列为请求的资源名称，Status 为请求的状态码。状态码和 Web 开发中的状态码是一致的，200 为成功请求，常见请求状态码如表 2-1 所示。Type 为此资源的类型、图片、脚本或者其他类型的静态文件类型。其他还包括了请求地址、文件大小以及请求时间等属性，供调试相应的属性。

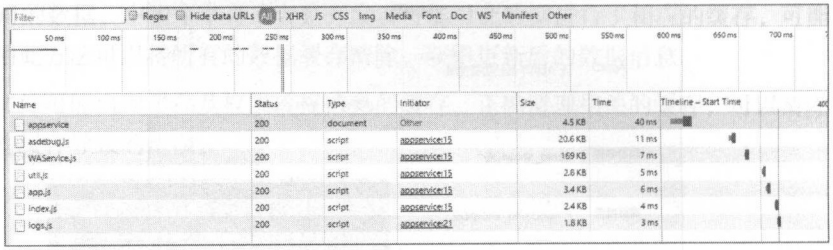


图 2-18　资源载入情况

表 2-1　常见请求状态码

常见状态码	说　明
200	服务器成功返回网页
304	缓存重定向
404	请求的资源不存在
503	服务器暂时不可用
500	服务器暂时不可用

注意：不只是初始化界面时显示的属性，在属性标题栏单击右键，可以对更多的属性进行查看。单击“属性标题栏”按钮可以对相关的所有请求进行排序，可以根据载入时间等优化程序，所有可以调试的属性如图 2-19 所示。

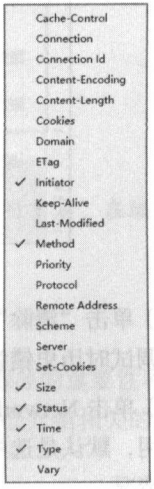


图 2-19　所有可以调试的属性

（5）单击调试窗口 Network 选项卡之后的“更多菜单”，可以调出其他调试菜单选项，如图 2-20 所示。

（6）单击 Storage 选项卡，可以看到现在的数据缓存。这里显示的数据缓存都是在小程序运行时产生的数据缓存，对于这些数据缓存，必要的时候可以进行清除和更新。需要清除缓存的时候可以单击左侧界面的“缓存”按钮，选择清除数据缓存，即可删除相关的数据，如图 2-21 所示。

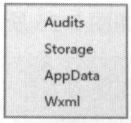


图 2-20 其他调试菜单选项

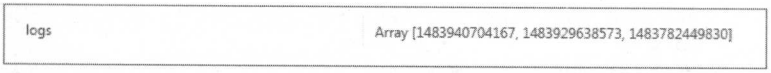


图 2-21 数据缓存

2.1.5 小程序的控制台

小程序开发工具的控制台是 JS 的调试工具，也是非常重要的代码调试工具和功能性调试工具，通过在控制台打印一些数据信息，可以对程序功能进行相关的调试，如图 2-22 所示。

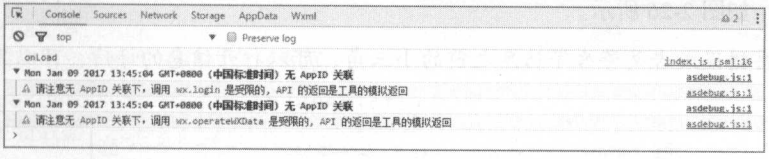


图 2-22 控制台信息

注意：打印控制台会自动对程序进行监测，除了程序本身的错误，有开发者的打印输出均会自动显示出代码的追踪栈。

开发者可以使用控制台进行调试操作。初始化小程序 Hello World 的时候，小程序的调试界面会自动显示两条不同的警告错误，通知开发者开发工具 API 为工具的模拟返回，而不是真实值，但是警告本身是不会中断程序的运行的。

可以人为对代码添加一个错误，单击“编辑”按钮，进入小程序的编辑代码页面，选择打开 pages 文件夹，进入 index.js 中，对其代码进行编辑。

这里人为制造一个语法错误——将程序的最后一行加上注释，使得整个代码区块不完整，如图 2-23 所示，以这样的方式来人为制造出控制台的报错信息。

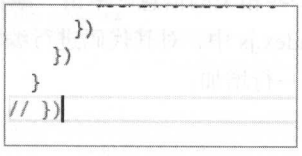


图 2-23 人为制造错误

这样会使得整个程序出现编译错误，按 Ctrl+s 组合键保存，这时小程序开发工具会出现一个提示栏，告知开发者代码编译出现错误，可以在控制台查看相应的详细报错信息，如图 2-24 所示。

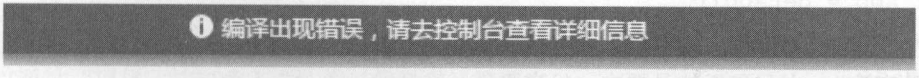


图 2-24 错误提示

再次单击调试窗口，可以看到调试页面下的控制台出现了两条错误信息，如图 2-25 所示。

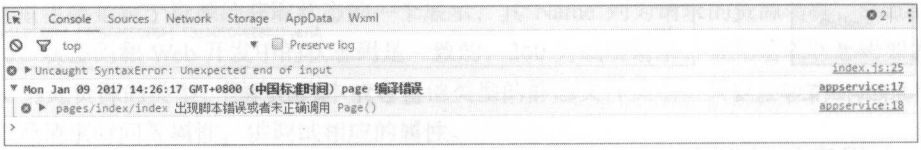


图 2-25　控制台错误提示

查看错误堆栈，可以看出第一条错误，右边显示的是错误的位置和代码行数，单击该处，会自动进入调试窗口的 Sources 选项卡，选项卡会在中间自动打开发生错误的文件，并且会自动标识发生错误的位置，如图 2-26 所示。

注意：单击错误或者是警告等信息之前的小三角，可以打开信息的详情，可以显示相关的错误堆栈等信息。

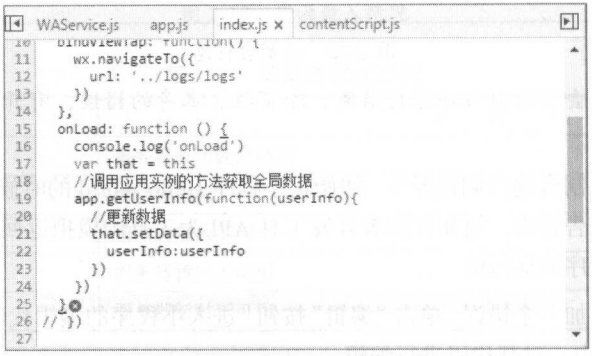


图 2-26　错误位置

这里出现了错误之后，整个小程序会自动中断运行，测试显示界面中不会有任何显示。

将错误修复后，可以尝试打印一些相关的信息。单击“编辑”按钮，进入小程序的编辑代码页面，选择打开 pages 文件夹，进入 index.js 中，对其代码进行编辑，先修复之前的错误，将最后一行的注释删除，并在 onLoad 函数最后一行增加：

```
console.log(userInfo)
```

则整个 index.js 的代码如下：

```
//index.js
//获取应用实例
var app = getApp()
Page({
  data: {
    motto: 'Hello World',
```

```
    userInfo: {}
  },
  //事件处理函数
  bindViewTap: function() {
    wx.navigateTo({
      url: '../logs/logs'
    })
  },
  onLoad: function () {
    console.log('onLoad')
    var that = this
    //调用应用实例的方法获取全局数据
    app.getUserInfo(function(userInfo){
      //更新数据
      that.setData({
        userInfo:userInfo
      })
      console.log(userInfo)
    })
  }
})
```

这里尝试打印开发者的信息，按 Ctrl+s 组合键保存，这时微信开发工具会自动重新加载小程序。再次单击“调试”按钮，可以发现控制台中已经打印出一些有关开发者的信息，如图 2-27 所示。

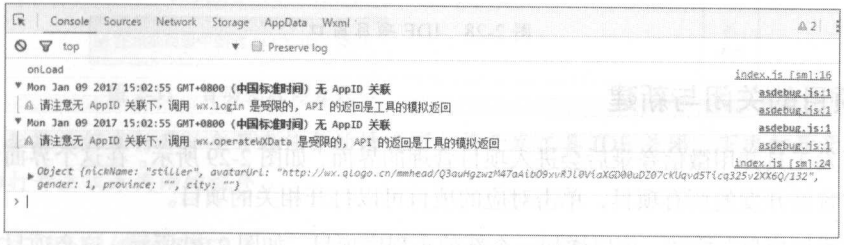


图 2-27 使用控制台进行打印调试

同样，单击打印结果的行数，可以跳转至 JS 代码中的 console.log(userInfo)代码处，也就是说，使用 console.log()方法可以输出结果在控制台中。

2.1.6 IDE 项目窗口

可以通过“项目”按钮进入微信小程序的项目管理窗口，主要是对于项目的一些管理属性和配

置信息。如果小程序本身已经关联了相关的 AppID，就可以在此界面上管理小程序的预览和上传，同时下方是对于项目本身的一些配置信息。

可以选择 JS 的代码是否支持 ES 6、代码补全、代码压缩，以及代码监听后的自动刷新功能等，如图 2-28 所示。



图 2-28　IDE 项目窗口

2.1.7　项目的关闭与新建

打开开发工具并用微信登录后会进入项目管理的界面，如图 2-29 所示。在这个界面中会显示出当前系统中曾经开发的所有项目，单击对应的项目可以打开相关的项目。

单击“添加项目”按钮，可以增加一个新的小程序项目，如图 2-30 所示。这个项目中存在三个可选项，其中 AppID 是通过微信公众号中的开发者申请的相关 ID，一个公众号暂时可以申请 5 个小程序 ID，而一个小程序则只能绑定到一个公众号中。如果只是做一个本地化的 Demo 或者没有条件获得相关的小程序，则可以通过选择无 AppID 来做本地开发。

注意：对于无 AppID 的情况下，选择开发小程序时是不能上传到相关的公众号中的，所以只能通过工具的模拟进行测试。

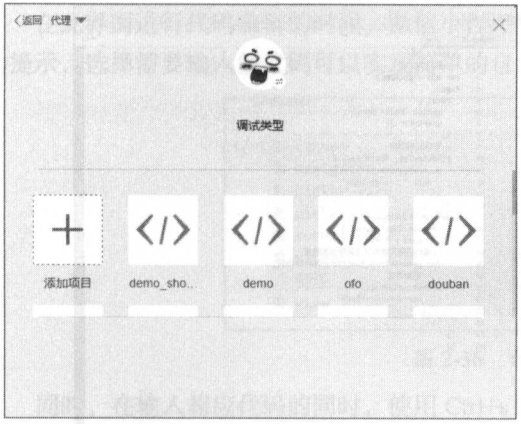


图 2-29 项目管理界面

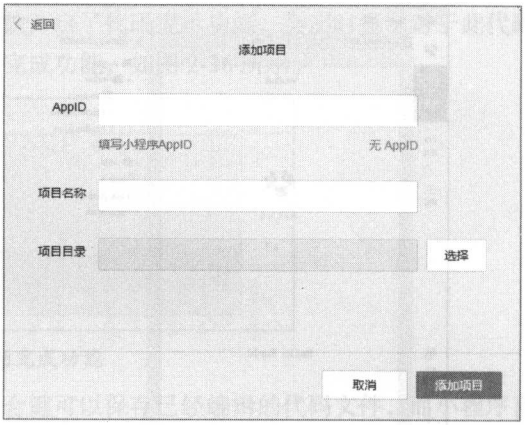


图 2-30 新建项目

选择合适的项目名称和项目目录，单击“添加项目”按钮，就可以建立起一个新的小程序应用，并且会自动打开代码编辑界面。当选择完相关的项目地址后，将会出现“在当前目录中创建 quick start 项目”的复选框，如图 2-31 所示。

如果勾选 quick start 选项，则会在新建的小程序中增加系统自带的 Hello World 应用，如果不勾选，则会新建一个空白应用。

在打开代码编辑界面的情况下，想要增加新的小程序应用，或者想关闭此开发任务，则需要单击“关闭”按钮将当前时间活跃的小程序关闭，即可以新建开发新的小程序或者打开其他的小程序开发环境。

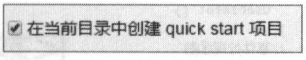


图 2-31 自动生成部分代码



图 2-32 关闭当前项目

注意：如果已经在打开一个开发任务的情况下，将开发工具 IDE 关闭，下次打开开发工具的时候，会自动打开上次未关闭的程序。

2.1.8 IDE 项目编辑窗口

单击左侧“编辑”按钮，可以进入小程序的编辑界面，如图 2-33 所示。

在编辑窗口中，可以对该小程序的代码进行编辑，除去小程序的展示部分，分为两个部分。

编辑界面的第一部分是文件管理部分，对于所有程序文件有展示作用，即文件管理，通过显示一个文件树来对所有的文件进行管理，如图 2-34 所示。

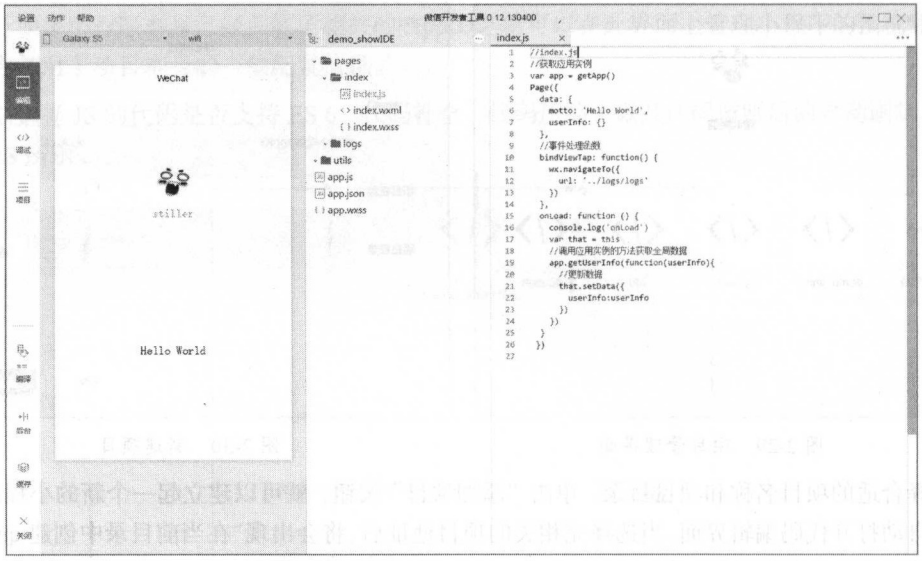


图 2-33　小程序编辑界面

单击想要编辑的文件，将会在旁边的代码编辑界面打开此文件，可以在编辑界面进行此文件的编辑。例如双击打开 index.js，具体界面如图 2-35 所示。

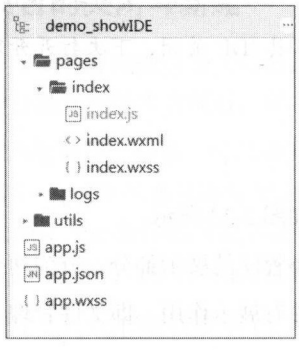


图 2-34　文件管理



图 2-35　index.js 界面

在此界面进行代码编辑的时候，微信小程序默认开启了代码提示功能，会实时显示对于此代码的提示，选择需要输入的代码可以实现简单的自动完成功能，如图 2-36 所示。

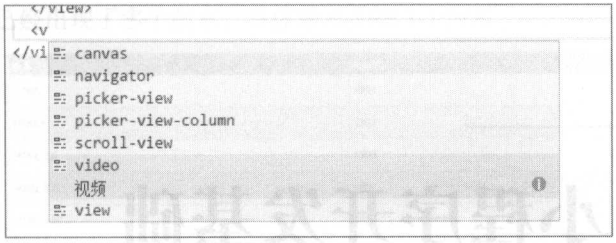


图 2-36 自动完成功能

同时，在输入相应代码的同时，使用 Ctrl+s 组合键可以保存已经编辑的代码文件，而小程序自身也将重启该程序。

注意：此功能如果不需要可以在项目管理界面关闭此功能，同时也可以关闭保存后自动重载的功能。

2.2 小结

本章通过对小程序开发工具的介绍，让读者了解小程序开发工具的使用方式和代码的调试方式。小程序开发工具是小程序开发的基本内容，熟练使用开发工具，可以帮助开发者简单和高效地开发微信小程序。由于小程序现在处于一个高速迭代的开发时期，所以开发工具的未来发展可能很快，会提供更多的开发调试、代码编辑及程序发布等功能。

2.3 实践与练习

- 熟悉本机安装的小程序开发环境。
- 对于小程序进行简单的调试，建立属于自己的小程序。

3 小程序开发基础

本章主要记录了一些微信小程序开发时所使用的技术的基础部分，系统介绍了网页开发HTML、CSS 及 JavaScript。因为这三种技术同时也是微信小程序开发中的基础，所以对于这三种技术的学习是非常重要的。

通过本章的学习，读者可以掌握：

- 基础的 JavaScript、CSS、HTML 语法。
- 基础的代码书写与编程规范。
- 常用的程序逻辑判断和基本读写代码的能力。

3.1 HTML 入门

通过对于本节的学习，读者可以掌握基础的 HTML 语法，理解 HTML 各种标签中的意义和使用场景，理解并可以书写 HTML 标签中的属性和设置属性的值。

3.1.1 HTML 简介

第 1 章中已经简单地介绍了 HTML 的一些历史和知识，这里主要介绍一下 HTML 的语言和标签。HTML 是用来描述网页的一种语言，全称是超文本标记语言（HyperText Markup Language），所以 HTML 不是一种编程语言，而是一种标记语言。

每一种标记语言都是由一套标记标签（markup tag）构成的，所以 HTML 也使用标记标签来描述网页，由标签构成的一个完整的 HTML 文档包含了 HTML 标签及文本内容，对于 HTML 而言，HTML 文档即 Web 页面。

HTML 标记标签通常被称为 HTML 标签（HTML tag），标签是由尖括号包围的关键词，比如

<html>。标签通常是成对出现的，比如<html>和</html>，一对标签构成了一个闭合标签。标签对中的第一个标签是开始标签，第二个标签是结束标签。

HTML 至今为止已经出现了多个版本，具体版本如图 3-1 所示。

版本	发布时间
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML 1.0	2000
HTML5	2012
XHTML5	2013

图 3-1 HTML 的版本

3.1.2 HTML 的用途与结构

对于新版的 HTML 5 类型的页面而言，其文档开头为：

```
<!DOCTYPE html>
```

这个表示也是现在的 HTML 文档的通用标识，也就是说，只要是在一个 HTML 文件中第一行显示的是这样一段标签代码，则证明其本身为一个标准的 HTML 文件。

以下是其他一些常用标签和意义：

- <!DOCTYPE html>声明为 HTML 5 文档。
- <html>元素是 HTML 页面的根元素。
- <head>元素包含了文档的元（meta）数据。
- <title>元素描述了文档的标题。
- <body>元素包含了可见的页面内容。
- <h1>元素定义一个大标题。
- <p>元素定义一个段落。

【示例 3-1】

一个完整的 HTML 文档页面：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>标准 HTML 文档</title>
```

```
</head>
<body>
<h1>H1 标题</h1>
<p>一个文字段落。</p>
</body>
</html>
```

新建一个 txt 文件，将扩展名改为.html，然后在其中书写上述代码，在浏览器中打开这个 HTML 文件，页面显示效果如图 3-2 所示。

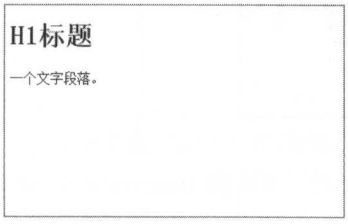


图 3-2　HTML 显示效果

3.2　HTML 基础

3.2.1　HTML 语法基础

HTML 的语法主要是通过 HTML 元素实现的，一个完整的 HTML 元素指的是从开始标签（start tag）到结束标签（end tag）的所有代码。

注意：开始标签常被称为开放标签（opening tag），结束标签常被称为闭合标签（closing tag）。

而对于 HTML 的元素语法有下面几点：

- HTML 元素以开始标签起始。
- HTML 元素以结束标签终止。
- 元素的内容是开始标签与结束标签之间的内容。
- 某些 HTML 元素具有空内容（empty content）。
- 空元素在开始标签中进行关闭（以开始标签的结束而结束）。
- 大多数 HTML 元素可拥有属性。

有些特殊的标签，不需要或者本身就不存在结束标签，称之为半闭合标签。

所有的 HTML 标签几乎都是可以嵌套的，通过不同的标签嵌套，即在一个 HTML 标签中可以包含其他的一个或者多个不同的 HTML 标签，并且成功显示。通过对于不同标签的嵌套，可以构成一个灵活的布局格式。

【示例 3-2】

标签的嵌套：

```
<html>
<body>
<p>这是段落标签</p>
```

```
</body>
</html>
```

【代码解析】

- 元素：

```
<p>这是段落标签</p>
```

元素定义了 HTML 文档中的一个段落，其拥有一个开始标签 ，以及一个结束标签 。

- <body>元素：

```
<body>
<p>这是段落标签</p>
</body>
```

<body>元素定义了 HTML 文档的主体，这个元素拥有一个开始标签<body>，以及一个结束标签</body>。元素内容是另一个 HTML 元素（p 元素）。

- <html>元素：

```
<html>
.....
</html>
```

<html>元素定义了整个 HTML 文档，这个元素拥有一个开始标签<html>，以及一个结束标签</html>。

上面的例子包含三个 HTML 元素，由一个<HTML></HTML>元素作为最底层的元素，在元素中嵌套了<body></body>元素，并且在<body></body>元素中嵌套了一个 标签。

一般即使忘记了使用结束标签，大多数浏览器也会正确地显示 HTML：

```
<p>这是段落标签
```

原本的效果依旧可以显示，没有任何影响，但不要依赖这种做法。忘记使用结束标签会产生不可预料的结果或错误。

注意：HTML 标签对大小写不敏感，也就是说对于 HTML 而言，无论是大写或者小写都可以正确解析。

3.2.2 HTML 元素

HTML 作为一种标记语言，存在很多既定的元素。

1. 标题标签（Heading）

标题标签是通过<h1>至<h6>标签进行定义的。<h1>定义最大的标题。<h6>定义最小的标题。

【示例 3-3】

标题标签：

```
<h1>h1</h1>
<h2>h2</h2>
<h3>h3</h3>
<h4>h4</h4>
<h5>h5</h5>
<h6>h6</h6>
```

显示效果如图 3-3 所示。

2. 水平线标签

用水平线标签<hr />在 HTML 页面中创建水平线，水平线可以用于切分不同的显示模块。

【示例 3-4】

```
<h1>水平线上方</h1>
<hr/>
<h1>水平线下方</h1>
```

其显示效果如图 3-4 所示。

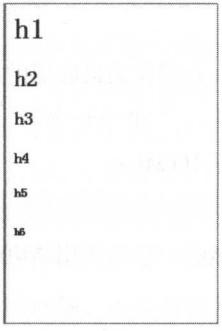


图 3-3　不同标题的显示效果

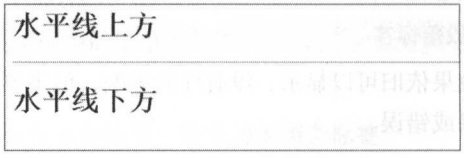


图 3-4　水平线显示效果

注意：如果显示的中文出现乱码，可能是因为编辑器的编码与显示器的编码不同而造成的。

3. HTML 注释

可以将注释插入 HTML 代码中，这样可以提高其可读性，使代码更易被人理解。浏览器会忽略注释，也不会显示它们。

HTML 注释写法为<!—这是注释的内容！-->，这段注释是不会被解析的，也就是说显示效果与其无关。

4. 段落标签

段落是通过 标签定义的，对于一组 标签中的文字信息，将会被解析器解析为一段文字。

【示例 3-5】

段落标签：

```
<p>
    这是段落中的文字。
    这是段落中的文字。
    这是段落中的文字。
    这是段落中的文字。
    这是段落中的文字。
</p>
```

其显示效果如图 3-5 所示。

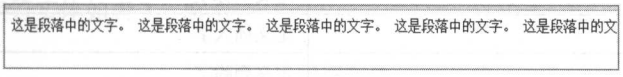


图 3-5 标签显示

5. 换行标签

换行标签
可是实现对于一段文字或信息、图片等元素的换行显示，如果希望在不产生一个新段落的情况下进行换行（新行），使用
标签。

【示例 3-6】

换行标签：

```
<p>
    这是段落中的文字。<br/>
    这是段落中的文字。<br/>
    这是段落中的文字。<br/>
    这是段落中的文字。<br/>
    这是段落中的文字。<br/>
</p>
```

其显示效果如图 3-6 所示。

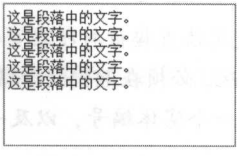


图 3-6 换行标签

6. 其他文字标签

文字标签中还存在很多修饰这些不同文字的标签，使得文字的显示效果不再单一，有利于突出重点，具体标签如表 3-1 所示。

表 3-1 文字标签与说明

标 签	说 明
<big>	定义大号字
	定义着重文字
<i>	定义斜体字
<small>	定义小号字
	定义加重语气
<sub>	定义下标字
<sup>	定义上标字
<ins>	定义插入字
<del>	定义删除字

【示例 3-7】

其他文字标签：

```
<p>
    <b>&lt;b&gt;定义粗体文本。</b><br/>
    <big>&lt;big&gt;定义大号字。</big><br/>
    <em>&lt;em&gt;定义着重文字。</em><br/>
    <i>&lt;i&gt;定义斜体字。<i><br/>
    <small>&lt;small&gt;定义小号字。<small><br/>
    <strong>&lt;strong&gt;定义加重语气。<strong><br/>
    <sub>&lt;sub&gt;定义下标字。<sub><br/>
    <sup>&lt;sup&gt;定义上标字。<sup><br/>
    <ins>&lt;ins&gt;定义插入字。<ins><br/>
    <del>&lt;del&gt;定义删除字。<del><br/>
</p>
```

显示效果如图 3-7 所示。

注意：HTML 页面中有一些符号是无法直接被打印在页面上的，因为它们可能被识别成有意义的标签的一部分，所以为了解决这种情况，必须在 HTML 源码中插入字符实体。字符实体有三部分：一个和号（&）、一个实体名称或者#、一个实体编号，以及一个分号（;）。要在 HTML 文档中显示小于号，需要写成<或者&#60。

使用实体名称而不是实体编号的好处在于，名称相对来说更容易记忆。而这么做的坏处是，并不是所有的浏览器都支持最新的实体名称，然而几乎所有的浏览器对实体编号的支持都很好。

字符实体对大小写敏感。

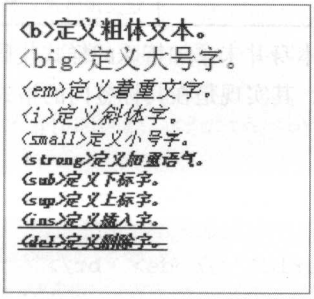


图 3-7 文字标签示例

其他经常使用的字符实体如表 3-2 所示。

表 3-2 字符实体

显示结果	说 明	实体名称	实体编号
	空格		
<	小于号、尖括号	<	<
>	大于号、尖括号	>	>
&	和号	&	&
"	引号	"	"
'	撇号	'	'
×	乘号	×	×
÷	除号	÷	÷
®	注册商标	®	®
©	版权	©	©
¥	日圆	¥	¥
£	镑	£	£
¢	分	¢	¢

7.“计算机输出”标签

可以格式化输出计算机中的代码、键盘码等特殊的文字格式，一般用于对于代码类型的文本式的输出，具体标签如表 3-3 所示。

表 3-3　计算机输出标签

标　签	说　明
<code>	定义计算机代码
<pre>	定义预格式文本

对于<code></code>标签而言，本身并未添加特殊的样式代码，只是将所有的文字变成了等宽的长度。而对于<pre></pre>标签而言，其实现是在其标签内的格式不会随着 HTML 的规则进行改变。

【示例 3-8】

“计算机输出”标签：

```
<p>
    <code>print "Hello World!"</code> <br/>
    <pre>
array(
    "a"=>"a",
    "b"=>"b"
    );
    </pre>
</p>
```

显示效果如图 3-8 所示。

```
print "Hello World!"

array(
        "a"=>"a",
        "b"=>"b"
        );
```

图 3-8　“计算机输出”标签显示效果

8. 引用、引用标签和术语标签

对于网页文章而言，很可能存在很多内容上的引用或者是对于论文的一些格式的信息，HTML 为了解决这个需求，拥有完整的一套引用标签，具体标签如表 3-4 所示。

表 3-4　引用标签

标　签	说　明
<abbr>	定义缩写或首字母缩略语
<address>	定义文档作者或拥有者的联系信息
<bdo>	定义文本方向
<blockquote>	定义从其他来源引用的节
<dfn>	定义项目或缩略词的定义
<q>	定义短的行内引用
<cite>	定义著作的标题

【示例 3-9】

术语标签：

```
<p>
    <abbr>&lt;abbr&gt;定义缩写或首字母缩略语。</abbr><br/>
    <address>&lt;address&gt;定义文档作者或拥有者的联系信息。</address> <br/>
    <bdo>&lt;bdo&gt;定义文本方向。</bdo><br/>
    <blockquote>&lt;blockquote&gt;定义从其他来源引用的节。</blockquote><br/>
    <dfn>&lt;dfn&gt;定义项目或缩略词的定义。</dfn><br/>
    <q>&lt;q&gt;定义短的行内引用。</q><br/>
    <cite>&lt;cite&gt;定义著作的标题。</cite><br/>
</p>
```

显示效果如图 3-9 所示。

<abbr>定义缩写或首字母缩略语。

<address>定义文档作者或拥有者的联系信息。

<bdo>定义文本方向。

<blockquote>定义从其他来源引用的节。

<dfn>定义项目或缩略词的定义。
"<q>定义短的行内引用。"
<cite>定义著作的标题。

图 3-9 引用标签示例

9. HTML 链接标签（<a>）

通过对于<a>标签的使用，可以轻松完成页面的跳转和资源的跳转功能，其跳转的地址被称为超链接。超链接可以是一个字、一个词，或者一组词，也可以是一幅图像，可以单击这些内容来跳转到新的文档或者当前文档中的某个部分。

对于 HTML 中的标准而言，通过<a>标签形成的超链接，如果把鼠标指针移动到网页中的某个链接上时，箭头变为一只小手，就是可以单击的状态。

对于<a>标签有两种使用的方式：

- 通过使用 href 属性：创建指向另一个文档的链接。
- 通过使用 name 属性：创建文档内的书签。

第一种使用方法是链接至其他页面的跳转式超链接，第二种使用方式则是在同一个页面中的锚点跳转，即是从当前浏览的位置页面会跳转至添加的锚点标签的位置。

【示例 3-10】

通过换行标签
使锚点标签超出页面显示，用于测试锚点链接。

```
单击这个a标签会跳转至百度<a href="http://www.baidu.com">BAIDU</a><br/>
单击这个链接会跳转至<a href="#maodian">锚点</a>
<br/><br/><br/><br/>
<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
<br/><br/><br/><br/><br/><br/><br/>
<a name="maodian">锚点</a>
```

显示效果如图 3-10 所示。

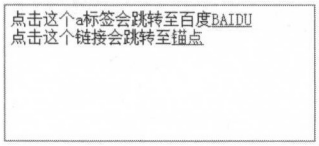


图 3-10 ＜a＞标签示例

注意：<a>元素最重要的属性是 href 属性，它指示链接的目标。在所有浏览器中，链接的默认外观是：

- 未被访问的链接带有下画线而且是蓝色的。
- 已被访问的链接带有下画线而且是紫色的。
- 活动链接带有下画线而且是红色的。

10. 图像标签（<img>）

在 HTML 中，图像由<img>标签定义的，用于对于 HTML 文档中需要的图片资源进行引入。<img>是半闭合标签，意思是说，它只包含属性，并且没有闭合标签。要在页面上显示图像，只需要使用源属性（src），src 指 source，它的值是图像的 URL 地址。

注意：对于<img>标签而言，是同时可以引用本地图片资源和互联网图片资源的。

【示例 3-11】

图像标签：

```
<img
src=http://file.digitaling.com/eImg/logo/20140127/20140127231635_73049_320.
png/>
```

显示效果如图 3-11 所示。

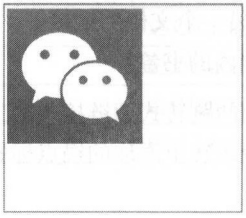


图 3-11 <img>标签显示示例

11. 表格标签（<table>）

表格标签是布局和数据显示时最常用的标签之一，通过对于数据或者是网页布局表格化操作，可以使整个布局变得更加整齐和合理。

对于表格标签而言，并不是像之前标签一样只拥有一组标签，其在<table></table>标签中需要包含<tr></tr>代表表格的一行，<td></td>代表一行中的一列。不仅如此，对于很多不同的前端框架，可能会使用到<thead></thead>作为表格的表头标签，<tbody></tbody>作为表格的表身标签。

【代码 3-12】

表格标签：

```
<table border="1">
    <thead>
        <tr>
            <td>表头列 1</td>
            <td>表头列 2</td>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>数据 1</td>
            <td>数据 2</td>
        </tr>
        <tr>
            <td>数据 3</td>
            <td>数据 4</td>
        </tr>
        <tr>
            <td>数据 5</td>
            <td>数据 6</td>
        </tr>
    </tbody>
</table>
```

显示效果如图 3-12 所示。

表头列1	表头列2
数据1	数据2
数据3	数据4
数据5	数据6

图 3-12 表格标签显示示例

12. HTML 列表标签

对于 HTML 中的列表，其书写方式和<table>标签的相似，不仅仅是由一个标签构成的，是由多个标签进行嵌套而显示的效果。列表标签分为无序列表、有序列表、自定义列表这三种不同的列表。

- 无序列表是一个项目的列表，此列项目使用粗体圆点（典型的小黑圆圈）进行标记。最底层的标签是<ul>标签。每个列表项为<ul>标签中嵌套的<li>标签。
- 有序列表也是一个项目的列表，但是列表项目使用数字进行标记。有序列表底层的标签是<ol>标签。每个列表项依旧是嵌套在<ol>标签中的<li>标签。
- 自定义列表不仅仅是一个项目的列表，而且是项目及其注释的组合。自定义列表以<dl>标签为最底层的标签。每个自定义列表项通过嵌套<dt>标签实现。每个自定义列表项的定义通过在<dt>标签下增加<dd>标签实现。

【示例 3-13】

三种列表的使用方式：

```
<h1>无序列表</h1>
<ul>
    <li>列表项 1</li>
    <li>列表项 2</li>
    <li>列表项 3</li>
    <li>列表项 4</li>
    <li>列表项 5</li>
</ul>
<h1>有序列表</h1>
<ol>
    <li>列表项 1</li>
    <li>列表项 2</li>
    <li>列表项 3</li>
    <li>列表项 4</li>
    <li>列表项 5</li>
</ol>
<h1>定义列表</h1>
<dl>
    <dt>列表项 1</dt>
    <dd>列表项 1 的介绍</dd>
    <dt>列表项 2</dt>
    <dd>列表项 2 的介绍</dd>
    <dt>列表项 3</dt>
```

```
    <dd>列表项 3 的介绍</dd>
    <dt>列表项 4</dt>
    <dd>列表项 4 的介绍</dd>
    <dt>列表项 5</dt>
    <dd>列表项 5 的介绍</dd>
</dl>
```

显示效果如图 3-13 所示。

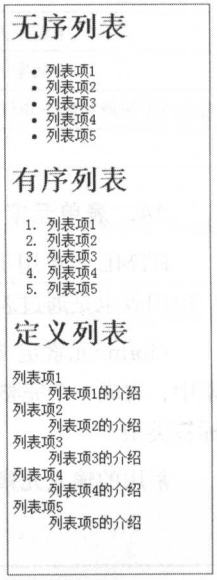


图 3-13 列表显示示例

13. HTML 块元素

HTML 块元素可以说是 HTML 中最常用的元素之一，通过对于块元素的定位，可以实现丰富的页面布局。 是最常见的块级元素，它是可用于组合其他 HTML 元素的容器。 元素没有特定的含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。如果与 CSS 一同使用， 元素可用于对大的内容块设置样式属性。

元素的另一个常见的用途是文档布局。它取代了使用表格定义布局的老式方法。使用<table>元素进行文档布局不是表格的正确用法。<table>元素的作用是显示表格化的数据。

一般对于基本的文字排版，也存在一对标签，元素也没有特定的含义，它也是内联元素，可用作文本的容器。当与 CSS 一同使用时，元素可用于为部分文本设置样式属性。

【示例 3-14】

div 的具体使用：

```
<h1>块元素的展示</h1>
<div style="height:100px;background-color:red">
    <p>这是第一块</p>
</div>
<div style="height:100px;background-color:blue">
    <p>这是第二块</p>
</div>
```

注意：通过对于块元素 的 style 属性的更新，将块元素的背景颜色和高度进行更改，使块元素的一些特征更容易展示。

显示效果如图 3-14 所示。

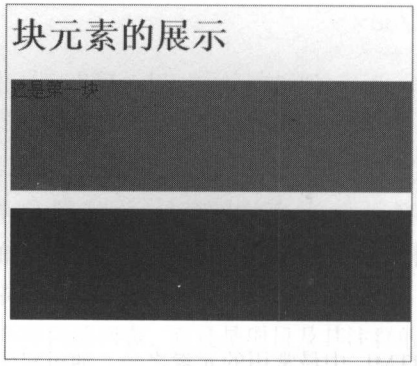


图 3-14　块元素显示效果

14．表单元素

HTML 表单用于收集用户输入，是网页与用户进行交互的重要组成部分之一，甚至于大量的接口应用等也是通过表单的方式进行获取和验证的。

<form>元素定义 HTML 表单，通过对于一些输入元素<input>标签的使用可以嵌套在<form>元素中，构成一个完整的表单。为了实现多种适合用户的输入模式，HTML 规定了几种不同的<input>标签类型。

常用的输入元素<input>的标签类型如表 3-5 所示。

表 3-5　<input>的标签类型

标签类型	说　明
text	定义常规文本输入
radio	定义单选按钮输入（选择多个选项之一）
submit	定义提交按钮（提交表单）
checkbox	定义复选框
Button	定义按钮

一些同样属于<input>标签的输入类型，但是一般不使用部分浏览器不适用的输入类型，如表 3-6 所示。

表 3-6　<input>标签的输入类型

标签类型	说　明
number	用于应该包含数字值的输入字段，能够对数字做出限制。根据浏览器支持，限制可应用到输入字段
date	用于应该包含日期的输入字段，根据浏览器支持，日期选择器会出现在输入字段中
color	用于应该包含颜色的输入字段，根据浏览器支持，颜色选择器会出现在输入字段中

续表

标签类型	说　明
range	用于应该包含一定范围的值的输入字段，根据浏览器支持，输入字段能够显示为滑块控件
month	允许用户选择月份和年份，根据浏览器支持，日期选择器会出现在输入字段中
week	允许用户选择周和年，根据浏览器支持，日期选择器会出现在输入字段中
time	允许用户选择时间（无时区），根据浏览器支持，时间选择器会出现在输入字段中
datetime	允许用户选择日期和时间（有时区），根据浏览器支持，日期选择器会出现在输入字段中
datetime-local	允许用户选择日期和时间（无时区），根据浏览器支持，日期选择器会出现在输入字段中
email	用于应该包含电子邮件地址的输入字段，根据浏览器支持，能够在被提交时自动对电子邮件地址进行验证
search	用于搜索字段（搜索字段的表现类似常规文本字段）
tel	用于应该包含电话号码的输入字段
url	用于应该包含 URL 地址的输入字段。根据浏览器支持，在提交时能够自动验证 url 字段。某些智能手机识别 url 类型，并向键盘添加".com"以匹配 url 输入

注意：对于不常用的标签类型，不建议在相关的需求中使用。相对于使用一些不被浏览器支持的类型，我更建议使用第三方开发者的插件来实现相关的功能。

【示例 3-15】

常见的表单及其<input>组件：

```
<h1>表单</h1>
<form>
    Text:<input type="text"><br/>
    Radio:<input type="radio"><br/>
    checkbox:<input type="checkbox"><br/>
    Button:<input type="button" value="Button"><br/>
    Submit:<input type="submit" value="Submit"><br/>
</form>
```

显示效果如图 3-15 所示。

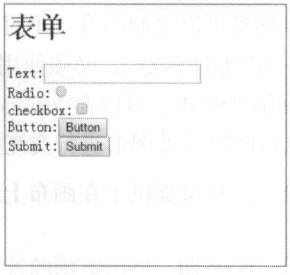


图 3-15　表单显示效果

15. 媒体标签

多媒体元素（比如视频和音频）存储于媒体文件中。确定媒体类型的最常用的方法是查看文件扩展名。当浏览器得到文件扩展名.htm 或.html 时，它会假定该文件是 HTML 页面。.xml 扩展名指示 XML 文件，而.css 扩展名指示样式表。图片格式则通过.gif 或.jpg 来识别。多媒体元素也拥有带有不同扩展名的文件格式，比如.swf、.wmv、.mp3 及.mp4。

<HTML>中可以通过媒体标签插入相应的媒体，完成比如在页面中引入一段视频或者是引入音频等操作。

<HTML>中 HTML 5 以前的媒体标签，对于引入一个媒体文件是比较复杂的，但是 HTML 5 新增了媒体<audio>、<video>标签，让媒体文件的引入变得更加方便和简单。

【示例 3-16】

媒体标签的引入和实现：

```
<h1>媒体标签</h1>
<div>
    <video src="movie.mp4" controls="controls">
    浏览器不支持 video 标签。
    </video>
</div>
<div>
    <audio autoplay="autoplay" src="music.mp3">
    浏览器不支持 audio 标签。
    </audio>
</div>
```

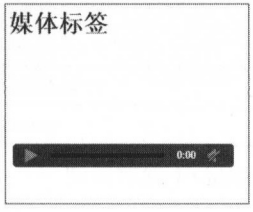


图 3-16　媒体标签显示效果

显示效果如图 3-16 所示。

16. 画布标签

画布标签<canvas>，也是 HTML 5 中的一个非常有用的标签，通过<canvas>标签可以绘制图像。不过，<canvas>元素本身并没有绘制能力（它仅仅是图形的容器），必须使用 JS 脚本来完成实际的绘图任务。通过对于画笔的操作，可以在<canvas>标签中绘制出不同的图案和图形，并且根据相关鼠标或者键盘事件可以对图案或者图形进行交互操作，甚至绘制出动画效果。

getContext()方法可返回一个对象，该对象提供了在画布上绘图的方法和属性。

【示例 3-17】

使用<canvas>标签：

```
<body>
    <canvas id="bg" width="500px" height="500px" style="border:1px solid
#000000;"></canvas>
</body>
<script>
    var c=document.getElementById("bg");
    var ctx=c.getContext("2d");
    ctx.beginPath();
    ctx.arc(200,250,90,0,2*Math.PI);
    ctx.stroke();
</script>
```

显示效果如图 3-17 所示。

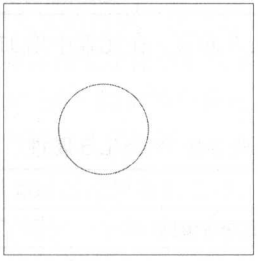


图 3-17 画布显示效果

3.2.3 HTML 属性

对于 HTML 标签中，单一的标签并不能完全表示某一种标签状态或者类型，所以所有的 HTML 标签中存在不同的属性。HTML 元素可以设置属性，标签属性可以在元素中添加附加信息，属性一般描述于开始标签，属性总是以名称/值（key=>value）对的形式出现，比如 name="value"。

对于任意标签来说，属性值应该始终被包括在引号内。一般使用双引号进行属性的书写，也可以使用单引号进行属性值的书写，同时支持单双引号的嵌套。

以<a>标签作为示例：

```
<a href="http://www.baidu.com">单击进入百度</a>
```

对于<a>标签而言，href 就是其标签中的一个属性。通过对于标签属性的添加，可以针对标签本身出现很多不同的显示效果和作用。

其他的一些例子：

将<h1>标签设置拥有关于对齐方式的附加信息，即将<h1>的对齐方式变成居中显示，具体代码如下：

```
<h1 align="center">
```

注意：属性和属性值对大小写不敏感。不过，万维网联盟在其 HTML 4 推荐标准中推荐小写的属性/属性值，而新版本的(X)HTML 要求使用小写属性。

对于所有的元素中存在一些常用的属性，这些属性值都是通用的，通用属性如表 3-7 所示。

表 3-7　通用属性

属　性	值	描　述
class	classname	规定元素的类名（classname）
id	id	规定元素的唯一 id
style	style_definition	规定元素的行内样式（inline style）
title	text	规定元素的额外信息（可在工具提示中显示）

除去 HTML 中的属性，对于 HTML 5 而言，在元素中增加了一些新的属性，一些常用的 HTML 5 属性如表 3-8 所示。

表 3-8　HTML 5 属性

属 性 名	描　述
contenteditable	规定元素内容是否可编辑
contextmenu	规定元素的上下文菜单，上下文菜单在用户单击元素时显示
data-*	用于存储页面或应用程序的私有定制数据
draggable	规定元素是否可拖动
dropzone	规定在拖动被拖动数据时是否进行复制、移动或链接
hidden	规定元素不再相关
spellcheck	规定是否对元素进行拼写和语法检查
translate	规定是否应该翻译元素内容

3.2.4　WXML 与 HTML

微信并不是直接使用 HTML 作为页面的开发语言，而是采用一种类 HTML 的标签语言作为开发语言，而这种标记语言被称为 WXML（Wei Xin Markup Language）。

因为 WXML 大体结构和属性与 HTML 一致，所以通过对于 HTML 的学习对 WXML 非常有帮助。

WXML 是框架设计的一套标签语言，结合基础组件、事件系统，可以构建出页面的结构。

注意：在 WXML 中，很多属性和书写方式与 HTML 都是一致的，所以对于 WXML 的一些写法可以参照 HTML。

3.3 CSS 入门

CSS 指层叠样式表，它的功能如下：

- 样式定义如何显示 HTML 元素。
- 样式通常存储在样式表中。
- 把样式添加到 HTML 中是为了解决内容与表现分离的问题。
- 外部样式表可以极大地提高工作效率。
- 外部样式表通常存储在 CSS 文件中。
- 多个样式定义可重复使用，并且只需要一处 CSS 样式。
- 样式解决了一个普遍的问题。

HTML 标签原本被设计为用于定义文档内容，而不是对于其样式进行更改。通过使用<h1>、<table>、<p>等不同含义的 HTML 标签来指定字段中的文字等内容的布局定位，比如某个字段是文章的标题，只需要将其包裹在<h1></h1>这一组标签中即可，无须明确指定其本身的布局。同时文档布局由浏览器来完成，而不使用任何的格式化标签。

样式表定义如何显示 HTML 元素，通过选择器的选择，将元素的样式与标签进行绑定。样式通常保存在外部的.css 文件中。通过仅仅编辑一个简单的 CSS 文档，外部样式表可同时改变站点中所有页面的布局和外观。

一个页面会多次引入不同的 CSS，而对于 CSS 中可能会存在重复的样式定义。当同一个 HTML 元素被不止一个样式定义时，样式表就会出现冲突，会存在先后级的引入。一般而言，样式应用的优先顺序如下：

- 内联样式（在 HTML 元素内部）。
- 内部样式表（位于<head>标签内部）。
- 外部样式表。
- 浏览器默认设置。

其中，内联样式（在 HTML 元素内部）拥有最高的优先权，这意味着它将优先于以下的样式声明：<head>标签中的样式声明、外部样式表中的样式声明浏览器中的样式声明（默认值）。

3.4 CSS 基础

本节通过介绍 CSS 的入门，引入 XWSS 的编辑和 CSS 的相似性和一致性，通过 CSS 的学习，可以让读者熟练掌握相关的 CSS 使用，并能引申到 WXSS 相关样式的编辑和显示。通过 CSS 的练习，让读者可以将其与 HTML 的基础进行结合，开发出多样的页面显示效果。

3.4.1 CSS 基础语法

CSS 规则由两个主要的部分构成：选择器和一个或多个声明。比如以下代码形式：

```
selector {declaration1; declaration2; ... declarationN }
```

selector 是一个元素的选择器，通过它可以指定相关的元素进行样式的修改。

1. 选择器

选择器用来改变 HTML 元素的 id 属性、class 属性、标签类型等。

【示例 3-18】

给一个 div 元素设定背景颜色的代码如下：

```
div{background-color:red}
```

在这个例子中，引入 CSS 文件的 HTML 中的所有 div 都会出现红色的背景，这个例子的完整代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>css_demo</title>
    <style>
        div{
            background-color: red;
        }
    </style>
</head>
<body>
<div>这是一个红色背景的 DIV</div>
</body>
</html>
```

具体显示效果如图 3-18 所示。

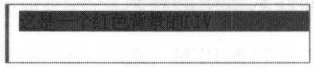


图 3-18　显示效果

2. 声明

每个声明都由一个属性和一个值组成。属性（property）是希望设置的样式属性（style attribute）。每个属性都有一个值，属性和值之间用冒号分开，如下所示：

```
selector {property: value}
```

【示例 3-19】

下面代码的作用是将 p 元素内的文字颜色定义为红色，同时将字体大小设置为 20 像素。在这个例子中，p 是选择器，color 和 font-size 是属性，red 和 20px 是值，完整的代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>css_demo2</title>
    <style>
        p {color:red; font-size:20px;}
    </style>
</head>
<body>
    <p>红色的 p 元素中的文字，20 号</p>
</body>
</html>
```

显示效果如图 3-19 所示。

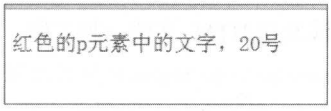


图 3-19 更改文字颜色和样式示例

注意：对于任意的样式表中，选择器后一定要使用花括号来包围声明，才符合正确的语法规范。

对于颜色值的不同写法和单位，除了英文单词 red，还可以使用十六进制的颜色值#ff0000：

```
p {color: ff0000; font-size:20px;}
```

以上的一段代码可以实现与上例相同的显示效果，为了节约字节，也可以通过两种方法使用 RGB 值：

```
p { color: rgb(255,0,0); font-size:20px;}
p { color: rgb(100%,0%,0%); font-size:20px; }
```

注意：当使用 RGB 百分比时，即使当值为 0 时也要写百分比符号。但是在其他的情况下就不需要这么做了。比如，当尺寸为 0 像素时，0 之后不需要使用 px 单位，因为无论单位是什么，都是 0。

3. 多个声明

如果要定义多个样式声明，则需要用分号将每个声明分开，但是最后一条规则是不需要加分号的。然而，在每条声明的末尾都加上分号，是一个好习惯。这么做的好处是，当从现有的规则中增

减声明时，会减少出错的可能性。

```
p {text-align:center; color:red; font-size:20px;}
```

每行只描述一个属性，这样可以增强样式定义的可读性，部分 IDE 会自动格式化代码，就像这样：

```
p {
  text-align: center;
  color: red;
  font-size:20px;
}
```

大多数样式表包含不止一条规则，而大多数规则包含不止一个声明。多重声明和空格的使用使得样式表更容易被编辑，并且能增加代码的可读性，比如：

```
body {
  color: #000;
  background: #fff;
  margin: 0;
  padding: 0;
  font-family: Georgia, Palatino, serif;
  }
```

样式代码中是否包含空格不会影响 CSS 在浏览器的工作效果。与 HTML 不同，CSS 对大小写不敏感。不过存在一个例外：如果涉及与 HTM 文档一起工作，class 和 id 名称对大小写是敏感的。

3.4.2 CSS 样式

CSS 允许应用纯色作为背景，也允许使用背景图像创建相当复杂的效果。CSS 在 HTML 中实现了更多的功能。

1. 背景色

可以使用 background-color 属性为元素设置背景色，它接受任何合法的颜色值，包括 RGB 和英文颜色。

【示例 3-20】

把两个 div 元素的背景设置为不同的两种颜色。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>css_demo</title>
    <style>
```

```
        .div1{
            background-color: red;
        }
        .div2{
            background-color: blue;
        }
    </style>
</head>
<body>
<div class="div1">这是一个红色背景的 DIV</div>
<div class="div2">这是一个蓝色背景的 DIV</div>
</body>
</html>
```

本例的显示效果如图 3-20 所示。

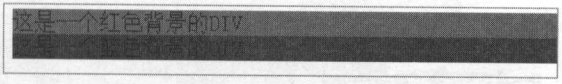


图 3-20 背景色显示效果

对于背景色的设置是元素的共同属性，可以为所有元素设置背景色，这包括 body、em 和 a 等行内元素。

background-color 不能继承，其默认值是 transparent。transparent 即为“透明”的意思。也就是说，如果一个元素没有指定背景色，那么背景默认是透明的，这样其上级元素的背景才能可见。

2. 背景图像

除了设置单一的背景色，同样也可以设置背景图像。要把图像放入背景，需要使用 background-image 属性。background-image 属性的默认值是 none，表示背景上没有放置任何图像，即默认是透明背景。如果需要设置一个背景图像，必须为这个属性设置一个 URL 值。

【示例 3-21】

对 body 元素设置一个背景图片，图片来自百度。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>css_demo4</title>
    <style>
        body{
            background-image: url("http://a-ssl.duitang.com/ uploads/item/
```

```
201203/29/20120329114141_Y3eQz.jpeg");
        }
    </style>
</head>
<body>
</body>
</html>
```

上方的代码实现，就相当于在 body 元素下设置了一张图片为背景，具体显示效果如图 3-21 所示。

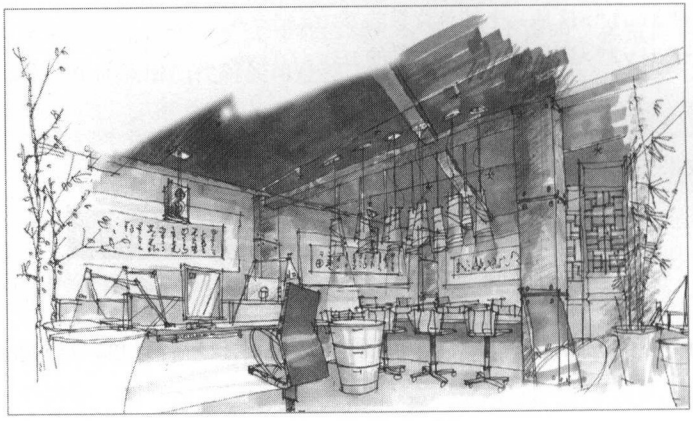


图 3-21　设置背景图示例

理论上讲，甚至可以向 textareas 和 select 等替换元素的背景应用图像，不过并不是所有元素都适合于增加背景图片。

注意：background-image 也不能继承。事实上，所有背景属性都不能继承，也就是说父元素上的背景仅属于父元素，子元素不会集成这个样式。

如果需要在页面上对背景图像进行平铺，可以使用 background-repeat 属性。其属性值 repeat 导致图像在水平和垂直方向上都平铺，就像以往背景图像的通常做法一样。repeat-x 和 repeat-y 分别导致图像只在水平或垂直方向上重复，no-repeat 则不允许图像在任何方向上平铺。默认背景图像将从一个元素的左上角开始，并且对于图像不做拉伸、重复或者实时缩放等处理。

背景图像的位置用 background-position 属性，它提供了很多值。首先，可以使用一些关键字：top、bottom、left、right 和 center。通常，这些关键字会成对出现，不过也不总是这样。还可以使用长度值，如 100px 或 5cm，也可以使用百分数值。不同类型的值对于背景图像的放置稍有差异。

图像放置关键字最容易理解，其作用如其名称所示。例如，top right 使图像放置在元素内边距区的右上角。根据规范，位置关键字可以按任何顺序出现，只要保证不超过两个关键字：一个对应水平方向，另一个对应垂直方向。如果只出现一个关键字，则认为另一个关键字是 center。

【示例 3-22】

如果希望每个段落的中部上方出现一个图像，只需做如下声明：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
    <style>
        div {
            height: 500px;
            width: 500px;
            background-image: url('http://a-ssl.duitang.com/ uploads/item/
201203/29/20120329114141_Y3eQz.jpeg');
            background-repeat: no-repeat;
            background-position: top;
        }

    </style>
</head>
<body>
<div></div>
</body>
</html>
```

其显示效果如图 3-22 所示。

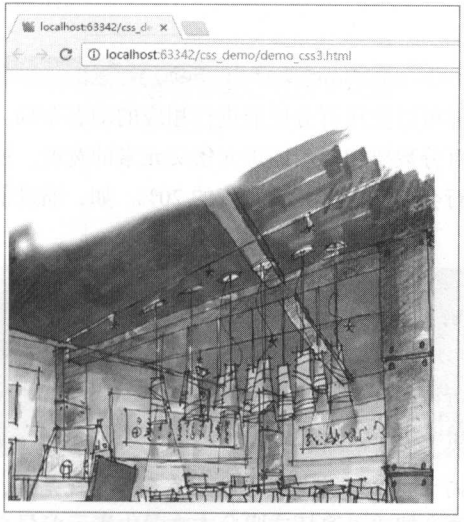


图 3-23 显示效果

3. 文本样式——缩进

通过 CSS 文本属性可定义文本的外观，并且对于文本本身的样式进行调整和优化，使整个文本呈现不同的显示效果，如文本的颜色、字符间距、对齐文本、装饰文本，对文本进行缩进可以使页面上段落的第一行缩进，这是一种最常用的文本格式化效果。

CSS 提供了 text-indent 属性，该属性可以方便地实现文本缩进。通过使用 text-indent 属性，所有元素的第一行都可以缩进一个给定的长度，甚至该长度可以是负值。下面的规则会使所有段落的首行缩进 5em：

```
p {text-indent: 5em;}
```

注意：一般来说，可以为所有块级元素应用 text-indent，但无法将该属性应用于行内元素，图像之类的替换元素上也无法应用 text-indent 属性。不过，如果一个块级元素（比如段落）的首行中有一个图像，它会随该行的其余文本移动。

如果想把一个行内元素的第一行“缩进”，HTML 中的文本并不具有太多复杂的样式，但是通过文本上属于父元素的样式进行调节，就可以用左内边距或外边距创造这种效果。

text-indent 还可以设置为负值。利用这种技术，可以实现很多有趣的效果，比如“悬挂缩进”，即第一行悬挂在元素中余下部分的左边：

```
p {text-indent: -5em;}
```

不过在为 text-indent 设置负值时要当心，如果对一个段落设置了负值，那么首行的某些文本可能会超出浏览器窗口的左边界。为了避免出现这种显示问题，建议针对负缩进再设置一个外边距或一些内边距：

```
p {text-indent: -5em; padding-left: 5em;}
```

文本除了绝对定位，同样可以使用百分比值进行相应的动态布局。即 text-indent 可以使用所有长度单位，包括百分比值。百分数要相对于缩进元素父元素的宽度。换句话说，如果将缩进值设置为 20%，所影响元素的第一行会缩进其父元素宽度的 20%。如，缩进值是父元素的 20%，即 100 像素：

```
div {width: 500px;}
p {text-indent: 20%;}

<div>
<p>this is a paragragh</p>
</div>
```

注意：对于文本的样式而言，其样式是存在继承关系的，就是说，通过父元素上设置的相关样式属性，其子元素同时会继承此样式属性。

【示例 3-23】

以下例子显示了继承关系，其内部<p>元素和 id 为 inner 的文字显示为相互对齐的方式。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
    <style>
        #outer {width: 500px;}
        #inner {text-indent: 10%;}
        p {width: 200px;}
    </style>
</head>
<body>
<div id="outer">
    <div id="inner">这个是 inner
        <p>内部 p 元素</p>
    </div>
</div>
</body>
</html>
```

具体的显示效果如图 3-23 所示。以上标记中的段落也会缩进 50 像素，这是因为这个段落继承了 id 为 inner 的 div 元素的缩进值。

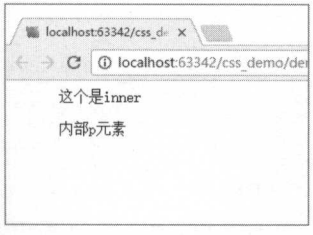


图 3-23 样式继承

4. 文本样式——text-align

text-align 是一个基本的文本属性，它会影响一个元素中的文本行互相之间的对齐方式。它有 5 个属性值：left、right、center、word-spacing、letter-spacing。前 3 个值相当直接，第 4 个和第 5 个则略有些复杂。

值 left、right 和 center 会导致元素中的文本分别左对齐、右对齐和居中。

注意：西方语言都是从左向右读，所以 text-alig 的默认值是 left。文本在左边界对齐，右边界呈锯齿状（称为“从左到右”文本）。对于希伯来语和阿拉伯语之类的语言，text-align 则默认为 right，因为这些语言从右向左读。不出所料，center 会使每个文本行在元素中居中。

text-align:center 与<CENTER>两种居中方式的区别如表 3-9 所示。

表 3-9　两种居中方式的区别

名　称	说　明
text-align:center	不会控制元素的对齐，而只影响内部内容。元素本身不会从一端移到另一端，只是其中的文本会受影响
<CENTER>	不仅影响文本，还会把整个元素居中

word-spacing 属性可以改变字（单词）之间的标准间隔。其默认值为 0，接受一个正长度值或负长度值。如果提供一个正长度值，那么字之间的间隔就会增加。为 word-spacing 设置一个负值，会把间距变窄。

【示例 3-24】

word-spacing 属性：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        p.ws1 {
            word-spacing: 30px;
        }

        p.ws2 {
            word-spacing: -10px;
        }
    </style>
</head>
<body>
<p class="ws1">
    This is the word-spacing = 30 px
</p>
<p class="ws2">
    This is the word-spacing = -10 px
</p>
```

```
</body>
</html>
```

具体显示效果如图 3-24 所示。

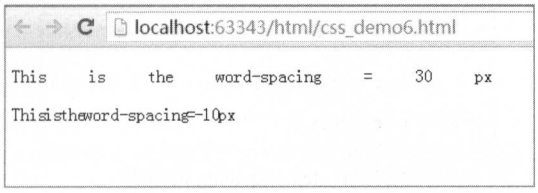


图 3-24 word-spacing 的显示效果

注意：word-spacing 属性对中文是无效的，只能用英文单词进行测试。

letter-spacing 属性则是字母与字母的间隔修改。与 word-spacing 属性一样，其属性的可取值包括所有长度。默认关键字是 normal（这与 letter-spacing:0 相同）。输入的长度值会使字母之间的间隔增加或减少指定的量。

【示例 3-25】

letter-spacing 属性：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        p.ls1 {
            letter-spacing: 10px;
        }

        p.ls2 {
            letter-spacing: -1px;
        }
    </style>
</head>
<body>
<p class="ls1">
    letter-spacing = 10 px
</p>
<p class="ls2">
    letter-spacing = -1 px
</p>
```

```
</body>
</html>
```

本例效果如图 3-25 所示。

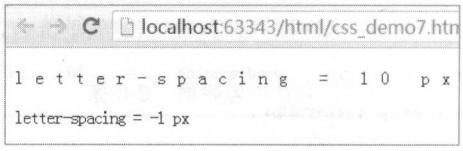


图 3-25　letter-spacing 的效果

5．字符转换

text-transform 属性用于处理文本的大小写，它有 4 个值，如表 3-10 所示。

表 3-10　大小写属性与说明

属　性	说　明
none	默认值 none 对文本不做任何改动，使用源文档中的原有大小写
uppercase	uppercase 将文本转换为全部大写
lowercase	lowercase 和全部小写字符
capitalize	capitalize 只对每个单词的首字母大写

注意：text-transform 属性的主要用于 CSS 中决定将所有的标题或者是其他的元素统一更改大小写时。

6．文本装饰

文本装饰使用 text-decoration 属性，可以对文字进行一些样式上的修改和增加一些新的显示效果，具体的属性如表 3-11 所示。

表 3-11　文字装饰属性

属　性	说　明
none	默认使用 none 对于文字样式默认无更改
underline	使用 underline 会对标签元素加下画线，与 HTML 中的 U 元素一样
overline	overline 的作用恰好相反，会在文本的顶端画一个上画线
line-through	line-through 则在文本中间画一个贯穿线，等价于 HTML 中的 S 和 strike 素
blink	blink 会让文本闪烁（只支持火狐浏览器）

【示例 3-26】

文本装饰：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        p.style1{
            text-transform:none;
            text-decoration:none;
        }
        p.style2{
            text-transform:uppercase
        }
        p.style3{
            text-transform:lowercase
        }
        p.style4{
            text-transform:capitalize
        }
        p.style5{
            text-decoration:underline
        }
        p.style6{
            text-decoration:overline
        }
        p.style7{
            text-decoration:line-through
        }
    </style>
</head>
<body>
<p class="style1">hello world</p>
<p class="style2">hello world</p>
<p class="style3">HELLO WORLD</p>
<p class="style4">hello world</p>
<p class="style5">Hello World</p>
<p class="style6">Hello World</p>
<p class="style7">Hello World</p>
</body>
</html>
```

显示效果如图 3-26 所示。

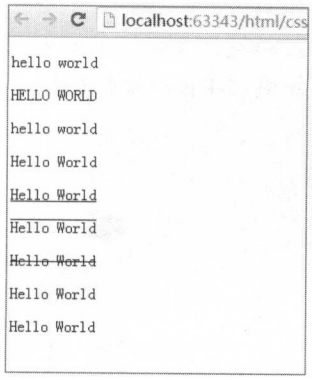


图 3-26　文本装饰效果

7．字体

CSS 的字体属性可以定义文本的字体系列、大小、加粗、风格（如斜体）和变形（如小型大写字母）等。CSS 使用 font-family 属性定义文本的字体。对于使用通用字体，通过制定 font-family 的属性可以使文档使用一种 sans-serif 字体。

通过以下代码可以制定 body 中的字体样式：

```
body {font-family: sans-serif;}
```

以上代码就会从 sans-serif 字体系列中选择一个字体（如 Helvetica），并将其应用到 body 元素。因为有继承，这种字体选择还将应用到 body 元素中包含的所有元素，除非有一种更特定的选择器将其覆盖。

注意：如果用户代理上没有安装特别指定的字体，就只能使用用户代理的默认字体来显示文字元素。

笔者建议在所有 font-famil 规则中都提供一个通用字体。这样在用户端未安装相应字体包的同时，依旧可以显示出默认的文字样式。

CSS 还可以进行字体风格的设置，通过更改 font-style 的属性值，以达到文本斜体、倾斜等效果，如表 3-12 所示。

注意：斜体（italic）和倾斜（oblique）的区别，italic 是一种简单的字体风格，对每个字母的结构有一些小改动，来反映变化的外观。与此不同， oblique 文本则是正常竖直文本的一个倾斜版本。

表 3-12　文字字体风格

属 性 值	说　明
normal	文本正常显示
italic	文本斜体显示
oblique	文本倾斜显示

CSS 也可以简单的实现字体加粗效果，类似于<srtong>标签，在 CSS 中使用 font-weight 属性设置文本的粗细，使用 bold 关键字可以将文本设置为粗体。

如果将元素的加粗设置为 bolder，浏览器会设置比所继承值更粗的一个字体加粗。与此相反，关键词 lighter 会导致浏览器将加粗度下移而不是上移。

通过以下代码就可以将所有的<p>元素中的文字都变为粗体：

```
p {font-weight:bold;}
```

当然，CSS 也可以控制字体的大小，通过设置 font-size 属性的数值可以设置文本的大小。通过以下代码就可以将所有<p>元素中的文字的字码都设为 14px：

```
p {font-size:14px;}
```

注意：大多数浏览器都可以通过浏览器的缩放工具来调整文本大小，但是这实际上是对整个页面的调整，而不仅限于文本。

【示例 3-27】

字体演示：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        p.normal {
            font-style:normal;
            font-weight:normal;
        }
        p.italic {font-style:italic;}
        p.oblique {font-style:oblique;}
        p.bold {font-weight:bold;}
        p {font-size:14px;}
    </style>
</head>
<body>
<p class="normal">Hello World</p>
<p class="italic">Hello World</p>
<p class="oblique">Hello World</p>
<p class="bold">Hello World</p>
</body>
</html>
```

文字字体样式示例如图 3-27 所示。

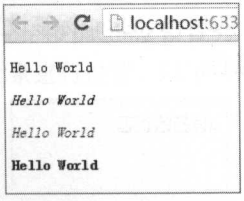


图 3-27 文字字体样式示例

8. 链接的样式

CSS 通过设置<a>标签的属性，可以设置链接的样式，例如 color、font-family、background 等。实现链接样式的特殊性在于，能够根据它们所处的状态来设置它们的样式（即当在此状态时而显示的效果）。链接的四种状态如表 3-13 所示。

表 3-13　链接的四种状态

状　态	说　明
a:link	普通的、未被访问的链接
a:visited	用户已访问的链接
a:hover	鼠标指针位于链接的上方
a:active	链接被单击的时刻

【示例 3-28】

链接样式：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        a:link {color:red;}       /* 未被访问的链接 */
        a:visited {color:blue;}   /* 已被访问的链接 */
        a:hover {color:yellow;}   /* 鼠标指针移动到链接上 */
        a:active {color:black;}   /* 正在被单击的链接 */

    </style>
</head>
<body>
<a href="http://www.baidu.com">a 标签的状态</a>
</body>
</html>
```

具体显示效果如图 3-28 所示。对于此样式的<a>标签设置，单击过后就会变成蓝色状态，鼠标悬停后显示黄色的标签，单击时变为黑色。



图 3-28　<a>标签状态示例

注意：当为链接的不同状态设置样式时，a:hover 必须位于 a:link 和 a:visited 之后，a:active 必须位于 a:hover 之后。

3.4.3 CSS 框模型

CSS 框模型 Box Model（也可以翻译为盒模型）规定了元素框处理元素内容、内边距、边框和外边距的方式，是 CSS 中非常重要的概念，框模型如图 3-29 所示。

元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景，内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

注意：背景应用于由内容和内边距、边框组成的区域。

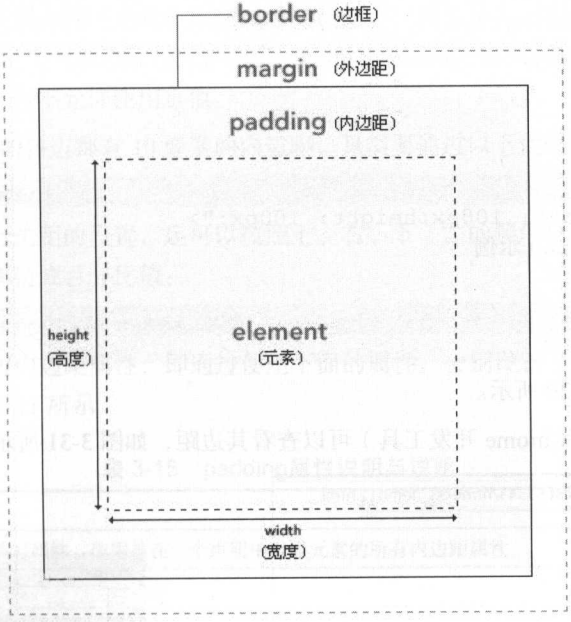


图 3-29 框模型

内边距、边框和外边距都是可选的，默认值是 0。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 margin 和 padding 设置为 0 来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器对所有元素进行设置。以下代码可以进行边距的更改：

```
* {
  margin: 0;
  padding: 0;
}
```

在 CSS 中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。

【示例 3-29】

CSS 框模型：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div{
            margin: 100px;
            padding: 100px;
        }
    </style>
</head>
<body>
<div style="width: 100px;height: 100px;">
    margin,padding 示例
</div>
</body>
</html>
```

框模型示例如图 3-30 所示。

通过测试工具（如 Chrome 开发工具）可以查看其边距，如图 3-31 所示。

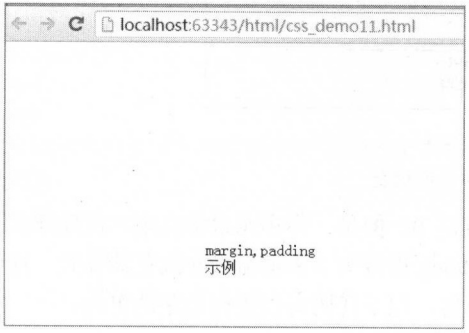


图 3-30　框模型示例

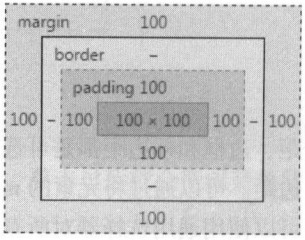


图 3-31　框模型的边距

注意：内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边距，外边距可以是负值，而且在很多情况下都要使用负值的外边距，这样可以让布局达到相应的效果。

根据 W3C 的规范，元素内容占据的空间是由 width 属性设置的，而内容周围的 padding 和 border 值是另外计算的。框模型元素的说明如表 3-14 所示。

表 3-14 框模型元素的说明

元 素	说 明
padding	内边距，也有资料将其翻译为填充
border	边框
margin	外边距，也有资料将其翻译为空白或空白边

注意：一般不要给元素添加具有指定宽度的内边距，而是尝试将内边距或外边距添加到元素的父元素和子元素中。

元素的内边距在边框和内容区之间。控制该区域最简单的属性是 padding 属性，对于 padding 属性而言，其值可以使用定义元素边框与元素内容之间的空白区域。padding 属性定义元素的内边距，接受长度值或百分比值，但不允许使用负值。

例如，对于 h1 元素的各边都有 10 像素的内边距，只需要通过以下代码实现：

```
h1 {padding: 10px;}
```

除了通过统一值进行边距的设置，还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

```
h1 {padding: 10px 0.25em 2ex 20%;}
```

当然也可以使用单边内边距属性，即通过使用下面的属性，分别设置上、右、下、左内边距，具体的属性与说明如表 3-15 所示。

表 3-15 padding属性说明与说明

属 性	描 述
padding	简写属性，作用是在一个声明中设置元素的所有内边距属性
padding-bottom	设置元素的下内边距
padding-left	设置元素的左内边距
padding-right	设置元素的右内边距
padding-top	设置元素的上内边距，元素的边框是围绕元素内容和内边距的一条或多条线

根据表 3-15 可以知道，对于 h1 {padding: 10px 0.25em 2ex 20%;}的代码效果和下面的规则实现的效果是完全相同的。

```
h1 {
  padding-top: 10px;
  padding-right: 0.25em;
  padding-bottom: 2ex;
```

```
    padding-left: 20%;
    }
```

CSS 可以设置边框，通过 border 属性可以规定元素边框的样式、宽度、颜色等。在 HTML 中，如果使用表格来创建文本周围的边框，那么通过 CSS 边框属性的设置，开发者可以创建出效果出色的边框，并且可以应用于任何元素。

元素外边距内就是元素的边框，元素的边框就是围绕元素内容和内边距的一条或多条线。每个边框有 3 个方面：宽度、样式、颜色，通过设置相关的属性可以对这 3 个方面进行设置。

1. 边框的样式

样式是边框最重要的一个方面，这不是因为样式控制着边框的显示（当然，样式确实控制着边框的显示），而是因为如果没有样式，将根本没有边框。

CSS 的 border-style（边框样式）属性定义了 10 个不同的非继承样式及 1 个继承样式，具体的属性如表 3-16 所示。

表 3-16　边框样式的属性

值	描　述
none	定义无边框
hidden	与 none 相同，不过应用于表时除外，对于表，hidden 用于解决边框冲突
dotted	定义点状边框，在大多数浏览器中呈现为实线
dashed	定义虚线，在大多数浏览器中呈现为实线
solid	定义实线
double	定义双线，双线的宽度等于 border-width 的值
groove	定义 3D 凹槽边框，其效果取决于 border-color 的值
ridge	定义 3D 垄状边框，其效果取决于 border-color 的值
inset	定义 3Dinset 边框，其效果取决于 border-color 的值
outset	定义 3Doutset 边框，其效果取决于 border-color 的值
inherit	规定应该从父元素继承边框样式

注意：可以为一个边框定义多个样式，其样式存在多个情况，相当于对 4 个边进行对应的设置。

当然也可以单独定义单边样式，为元素框的某一个边设置边框样式，而不是设置所有 4 个边的边框样式，可以使用表 3-17 的属性设置单边边框样式。

2. 边框的宽度

可以通过 border-width 属性为边框指定宽度。同样，设置相关的边框宽度也存在两种书写方式，可以单独设置边框的宽度，或者使用属性设置相关的样式，即可以使用按照 top-right-bottom-left 的

顺序设置元素的各边边框：

```
p {border-style: solid; border-width: 5px 5px 5px 5px;}
```

或者是通过表 3-18 中的属性分别设置边框各边的宽度。

表 3-17 不同方向样式

属 性 值	说 明
border-top-style	上边框样式
border-right-style	右边框样式
border-bottom-style	底边框样式
border-left-style	左边框样式

表 3-18 单独设置不同方向边框的宽度

属 性 值	说 明
border-top-width	上边框的宽度
border-right-width	右边框的宽度
border-bottom-width	底边框的宽度
border-left-width	左边框的宽度

3. 边框的颜色

border-color 属性可以设置边框的颜色，一次可以接受最多 4 个颜色值。可以使用任何类型的颜色值，例如可以是命名颜色，也可以是十六进制或者 RGB 值。对于一些单边边框颜色属性，它们的原理与单边样式和宽度属性相同，具体属性如表 3-19 所示。

表 3-19 单独设置不同方向边框的颜色

属 性 值	说 明
border-top-color	上边框的颜色
border-right-color	右边框的颜色
border-bottom-color	底边框的颜色
border-left-color	左边框的颜色

【示例 3-30】

边框使用：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title></title>
</head>
<style>
    h1 {
        border-style: solid;
        border-top-width:10px;
        border-color: black;
        border-right-color: red;
    }
</style>
<body>
    <h1>边框实例</h1>
</body>
</html>
```

上述代码的效果如图 3-32 所示。



图 3-32　边框示例

注意：围绕在元素边框的空白区域是外边距。设置外边距会在元素外创建额外的“空白”。

设置外边距最简单的方法就是使用 margin 属性，它接受任何长度单位（像素、英寸、毫米等）、百分数值甚至负值。同样 margin 属性也支持对于四个方向的内边距设置。如下代码就是对于<h1>标签的 4 个内边距进行了不同的设置：

```
h1 {margin : 10px 0px 15px 5px;}
```

与内边距的设置相同，这些值的顺序是从上外边距（top）开始围着元素顺时针旋转的，即其也是遵循 top、right、bottom、left 这样的顺序进行的。

注意：margin 属性的默认值是 0，所以如果没有为 margin 属性声明一个值，就不会出现外边距。但是，在实际中，浏览器对许多元素已经提供了预定的样式，外边距也不例外。

对于这种支持四个方向定义不同的 4 个值的属性，如果值不足 4 个，也是可以实现想要的效果的，其遵循的规则如下：

- 如果缺少左外边距的值，则使用右外边距的值。
- 如果缺少下外边距的值，则使用上外边距的值。
- 如果缺少右外边距的值，则使用上外边距的值。

当然，对于 margin 属性而言，也可以使用表 3-20 中任何一个属性来设置相应上的外边距，而不会直接影响所有其他外边距。

表 3-20 margin不同方向边距

属 性 名	说 明
margin-top	上方内边距
margin-right	右侧内边距
margin-bottom	下方内边距
margin-left	左侧内边距

一个规则中可以使用多个单边属性，例如：

```
p {
  margin-top: 10px;
  margin-right: 20px;
  margin-bottom: 30px;
  margin-left: 40px;
  }
```

当然，对于这种写法，其实现的效果和下方的代码效果是相同的。

```
p {margin: 20px 30px 30px 20px;}
```

不论使用单边属性还是使用 margin 属性，得到的结果都一样。一般来说，如果希望为多个边设置外边距，使用 margin 属性会更容易一些。不过，从文档显示的角度看，实际上使用哪种方法都不重要，所以应该选择对自己来说更容易的一种方法。

3.4.4 CSS 定位

CSS 定位（positioning）允许我们对元素进行定位或浮动显示，它提供的一些属性可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务，这种思想为 DIV+CSS 技术。

定位的基本思想很简单，它允许定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置，通过浮动、折叠等形式的实现，来完成一个独特的页面。

div、h1 或 p 元素常常被称为块级元素，这意味着这些元素显示为一块内容，即“块框”。与之相反，span 和 strong 等元素被称为“行内元素”，这是因为它们的内容显示在行中，即“行内框”。

在 CSS 中可以使用 display 属性改变生成的框的类型。这意味着，通过将 display 属性设置为 block，可以让行内元素（比如<a>元素）表现得像块级元素一样。还可以通过把 display 属性设置为 none，让生成的元素根本没有框。这样该框及其所有内容就不再显示，不占用文档中的空间。

CSS 有三种基本的定位机制：普通流、浮动和绝对定位。

- 除非专门指定，否则所有框都在普通流中定位。也就是说，普通流中元素的位置由元素在

HTML 中的位置决定。

- 块级框从上到下排列，框之间的垂直距离是由框的垂直外边距计算出来的。
- 行内框在一行中水平布置。可以使用水平内边距、边框和外边距调整它们的间距。但是，垂直内边距、边框和外边距不影响行内框的高度。由一行形成的水平框称为行框（LineBox），行框的高度足以容纳它包含的所有行内框。不过，设置行高可以增加这个框的高度。

通过使用 position 属性，可以选择 4 种不同类型的定位，这会影响元素框生成的方式。position 属性值和说明如表 3-21 所示。

表 3-21　position属性值和说明

属 性 值	说　明
static	元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中
relative	元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留
absolute	元素框从文档流完全删除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框
fixed	元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身

注意：相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

【示例 3-31】

position 属性：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
    <style type="text/css">
        .p1 {
            position: relative;
        }
        .p2 {
            position: static;
        }
        .p3 {
            position: absolute;
        }
        .p4 {
```

```
            position: fixed;
        }
    </style>
</head>

<body>
    <p class="p1">relative 例子</p>
    <p class="p2">static 例子</p>
    <p class="p3">absolute 例子</p>
    <p class="p4">fixed 例子</p>
</body>
</html>
```

Position 属性的显示效果如图 3-33 所示。

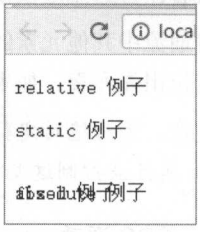


图 3-33 position 属性的显示效果

当然，HTML 中的元素也支持垂直排列图像，通过对于 Z-index 进行值的设置就可以简单地实现对于元素的覆盖效果。Z-index 可被用于将在一个元素放置于另一元素之后。对于 Z-index 而言，其最大值会出现在所有元素的最上层。

CSS 定位属性允许对元素进行定位，通过对于不同属性值的设定，可以达到不同的定位方式，具体的属性和说明如表 3-22 所示。

表 3-22 元素不同定位属性和说明

属　性	描　述
position	把元素放置到一个静态的、相对的、绝对的或固定的位置中
top	定义了一个定位元素的上外边距边界与其包含块上边界之间的偏移
right	定义了定位元素右外边距边界与其包含块右边界之间的偏移
bottom	定义了定位元素下外边距边界与其包含块下边界之间的偏移
left	定义了定位元素左外边距边界与其包含块左边界之间的偏移
overflow	设置当元素的内容溢出其区域时发生的事情
clip	设置元素的形状，元素被剪入这个形状之中，然后显示出来
vertical-align	设置元素的垂直对齐方式
z-index	设置元素的堆叠顺序

设置为相对定位的元素框会偏移某个距离。元素仍然保持其未定位前的形状，它原本所占的空间仍保留。

1. CSS 相对定位

相对定位是一个非常容易掌握的概念。如果对一个元素进行相对定位，它将出现在它所在的位

置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点进行移动。

注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其他框。

2. CSS 绝对定位

设置为绝对定位的元素框从文档流完全删除，并相对于其包含块定位，包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像该元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

绝对定位使元素的位置与文档流无关，因此不占据空间。这一点与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

普通流中其他元素的布局就像绝对定位的元素一样不存在。绝对定位的元素的位置相对于最近的已定位祖先元素，如果元素没有已定位的祖先元素，那么它的位置相对于最初的包含块。

注意：因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其他元素。可以通过设置 Z-index 属性来控制这些框的堆放次序。

3. CSS 浮动

CSS 通过 float 属性实现元素的浮动。对此属性进行值设定，可以实现元素的左浮动和右浮动。对于 float 属性而言，可以有 4 个相对应的值，实现不同的浮动效果，如表 3-23 所示。

表 3-23　float属性与说明

属 性 值	说　明
left	元素向左浮动
right	元素向右浮动
none	默认值，元素不浮动，并会显示在其在文本中出现的位置
inherit	规定应该从父元素继承 float 属性的值

【示例 3-32】

浮动演示：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
    <style type="text/css">
        .div1 {
            background-color: aqua;
```

```
            float: right
        }

        .div2 {
            background-color: red;
            float: left;
        }
    </style>
</head>

<body>
<h1>左浮动和右浮动</h1>
<p>
<div class="div1">右浮动的 div</div>
<div class="div2">左浮动的 div</div>
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
</body>

</html>
</body>
</html>
```

上述代码在一个<p>元素中增加了两个不同浮动方式的 标签，并且增加了文字作为填充。其显示效果如图 3-34 所示。



图 3-34 浮动显示效果

3.4.5 CSS 与 WXSS

对 CSS 的学习其实就是对 WXSS 的学习，虽然 WXSS 与 CSS 并不是完全相同的编辑方式，但是一些基础的属性都是一致的。

WXSS 的基础部分都符合 CSS 的语法和编辑方式，所以通过对于 CSS 的学习可以大体了解对于 HTML 元素的设置和布局方式，这对于微信小程序开发的理解是非常重要的。

3.5 JavaScript 入门

本节可以了解为什么选择 JavaScript 作为微信小程序开发的脚本语言，也可以了解 JavaScript 的诞生和至今的发展。

3.5.1 JavaScript 简介

JavaScript 是世界上最流行的编程语言，多年长居 GitHub 代码量的榜首，2016 年其开源项目数量远超出其他语言的开源项目数量，如图 3-35 所示。

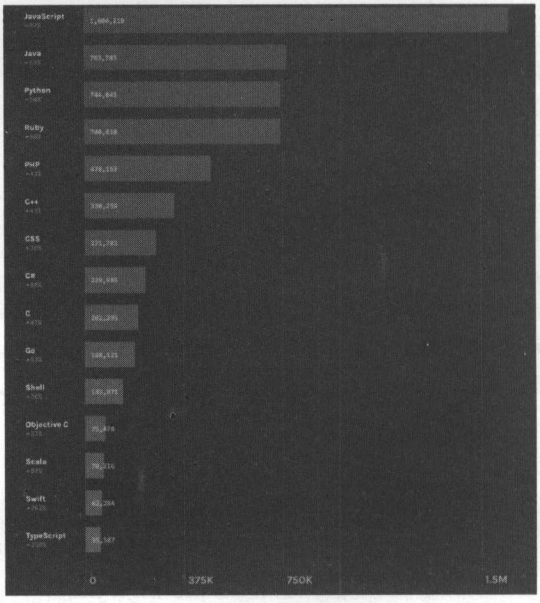


图 3-35 GitHub 上各项目占有率

这门语言不仅可用于 HTML 和 Web 开发，更广泛用于服务器、Mac、PC、笔记本电脑、平板电脑和智能手机等设备。

由于浏览器的支持，对于 JavaScript 这门语言而言，浏览器就是其最好的解析环境，所以其跨平台跨设备的特性也成了它之所以流行的原因之一。

JavaScript 具有以下特性：

- JavaScript 是脚本语言。
- JavaScript 是一种轻量级的编程语言。
- JavaScript 是可插入 HTML 页面的编程代码。
- JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。
- JavaScript 很容易学习。

正是因为 JavaScript 易学易用的特性，它在最近几年的时间里大放异彩，成为了整个编程界的新星，同时出现了大量基于 JavaScript 的开源项目。如今的 JavaScript 不仅仅可以用于浏览器去实现一些简单的动画或者是运算等效果，更可以运行于后端，为用户提供高并发、高性能的后端服务，比如 Node.js，甚至可以建立整个系统环境 OS.js。

3.5.2 JavaScript 前景

伴随着新版 ES 6/ES 2015 标准的出炉和开源项目的兴起，JavaScript 现在已经不仅仅只是对 HTML 中的 DOM 进行操作，而更像一门简单易学而功能强大的全能型语言，适用范围也不仅仅被局限于浏览器本身。现在的 JavaScript 的使用领域：

- 前端开发。
- 后端开发。
- Hybrid App。
- 桌面应用。
- React/React Native。
- 游戏引擎。
- 其他领域。

ES 6 标准的发布可谓是 JavaScript 历史上最重要的里程碑，它给 JavaScript 带来了诸多语言特性，箭头操作符、类的支持、字符串模板、函数参数默认值、迭代器、for-of 遍历、生成器、Symbols 基本类型等。在最近的两年中，出现了一批优秀的 JavaScript 开源项目，这些项目无疑推进了 JavaScript 的发展和更新，部分优秀的项目如图 3-36 所示。

所以对于 JavaScript 的前景是无须担心的，包括微信小程序在内，很多项目都选择了 JavaScript 作为其脚本开发语言，这也说明 JavaScript 的强大和应用广泛。

当然也正是因为这些原因，微信小程序选择了 JavaScript 作为自身的逻辑开发语言，通过简单

的 JavaScript 学习，读者便可以进行小程序的开发工作，入门门槛降低，对读者来说也是一件好事。

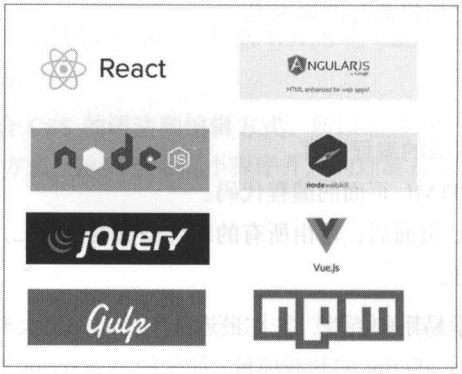


图 3-36　部分 JavaScript 开源项目

3.6　JavaScript 基础

通过对于本节的学习，可以了解基础的 JavaScript 书写规范和基础语法，包括了注释、变量、数据类型、对象、逻辑控制语句等基础的语法。

3.6.1　JavaScript 注释

注释是指编程中用于提高代码的可读性的说明部分。任何代码中，注释都是非常重要的组成部分，每一个注释都是对编程人员的一种指引和提示。开发者可以通过添加注释来提高代码的可读性。

JavaScript 支持单行注释和多行注释。单行注释以“//”开头，如：

```
//输出 DOM1：
document.getElementById("myDOM").innerHTML="这是 DOM1";
//输出 DOM2：
document.getElementById("myDOM 1").innerHTML="这是 DOM2";
```

多行注释用“/*……*/”实现，如：

```
/***创建时间
创建人  ***/
```

3.6.2　JavaScript 变量

变量是存储信息的容器。JavaScript 变量值用于数据的临时存储、计算、显示等。本节通过使用控制台中的 console.log()方法进行变量的一些输出测试。在 Chrome 系列浏览器（包括国产浏览器中

使用的 Chromium）中可以按 F12 键调出浏览器的控制台，对于此控制台的学习可以参照小程序调试一节。打开控制台后，显示效果如图 3-37 所示。

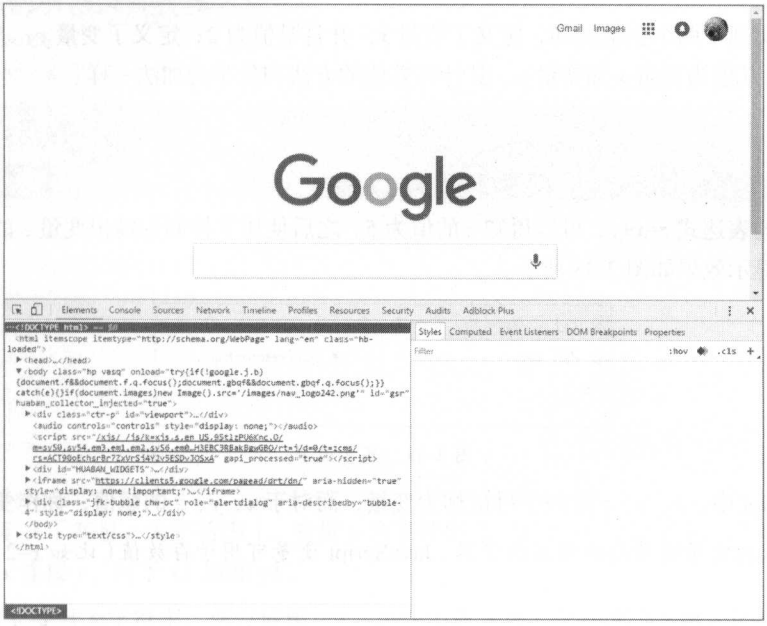


图 3-37 浏览器控制台

注意：默认打开控制台时会默认显示 Elements 选项卡，这个选项卡会显示当前页面的 HTML 代码，对于 JS 的测试输出结果需要选择 Console 选项卡。

【示例 3-33】

变量：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var x=2;
        var y=3;
        var z=x+y;
        console.log(z);
    </script>
</head>
```

```
<body>
</body>
</html>
```

上述代码实现了两个变量的和，定义了变量 x，并且赋值为 2；定义了变量 y，并且赋值为 3；定义了变量 z，其值为变量 x 加变量 y。其计算数值的方法和数学的加法一样。

```
x=2
y=3
z=x+y
```

通过上面的表达式 $z=x+y$，可以得知 z 的值为 5。之后使用了控制台输出变量 z 的数值，打印在控制台中。其显示效果如图 3-38 所示。

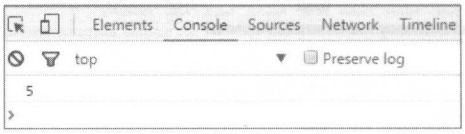


图 3-38　显示效果

在 JavaScript 中，x、y、z 这些字母被称为变量，而对于 x、y、z 的赋值，称作变量对应的值。

注意：可以把变量看作存储数据的容器。JavaScript 变量可用于存放值（比如 $x=2$）和表达式（比如 $z=x+y$）。

变量存在一定的命名规范：

- 变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age、sum、totalvolume）。
- 变量必须以字母开头。
- 变量也能以$和_符号开头。
- 变量名称对大小写敏感（y 和 Y 是不同的变量）。

除了数据以外，JavaScript 变量还能保存其他数据类型，比如文本值，这类变量的值称为字符串。对于值为字符串的变量，分配其文本值时，应该用双引号或单引号包围这个值，否则 JavaScript 会报语法错误。

对于字符串值，可以使用“+”进行连接，此时“+”不再作为“加法”，而是作为连接字符串的操作。

【示例 3-34】

字符串变量：

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var x="1";
        var y="2";
        var x1=1;
        var y1=2;
        var z=x+y;
        var z1=x1+y1;
        console.log(z);
        console.log(z1);
    </script>
</head>
<body>

</body>
</html>
```

上述代码定义了变量 *x* 为字符串 1，变量 *y* 为字符串 2。定义 x1 为数字 1，定义 y1 为数字 2，并且分别对于 *x* 连接 *y*，对于 x1 加和 y1。

由于 *x* 和 *y* 变量为字符串，所以使用“+”是作为连字符，合并两个字符串的操作，而非进行 1+2 的运算。所以对于 *z* 的输出是“1”连接“2”，结果为“12”。而其对比项是数字 1+2，输出变量 *z*1 为 3。

本例控制台打印结果如图 3-39 所示。

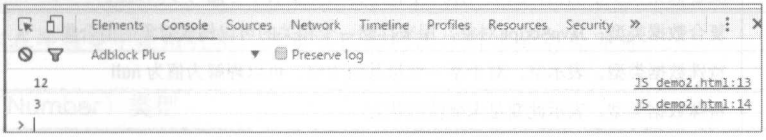


图 3-39 打印结果

注意：JavaScript 的变量取名不能用保留字，表 3-24 是 JavaScript 中的保留字。它们不能用于如 JavaScript 变量、函数、方法、循环标签、任何对象名称。

表 3-24 JavaScript中的保留字

abstract	boolean	break	byte
case	catch	char	class
const	continue	debugger	default
delete	do	double	else

续表

enum	Export	extends	false
final	finally	float	for
function	goto	if	implements
import	in	instanceof	int
interface	long	native	new
null	package	private	protected
public	return	short	static
super	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	

3.6.3 JavaScript 数据类型

JavaScript 的变量除了整数和字符串以外，还支持丰富的数据类型，这些类型如表 3-25 所示。

表 3-25 JavaScript数据类型

数据类型	说　明
String	基本数据类型，字符串，用于字符和字符串的显示
Number	基本数据类型，数字，用于数字的记录和运算，可以包含整数和浮点数
Boolean	基本数据类型，布尔值，包含 true 和 false 两种不同的值，用于逻辑判断
Array	复合数据类型，数组，数组对象用来在单独的变量名中存储一系列的值
Object	复合数据类型，JavaScript 对象。用来存储一个相关的可以构成对象的值合集
Null	特殊数据类型，表示空，对于某一变量其值为空，可以理解为值为 null
Undefined	特殊数据类型，表示此变量未赋值的状态

JavaScript 并不是强类型的语言，其拥有动态类型，这意味着相同的变量可用作不同的类型。

【示例 3-35】

动态类型：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
```

```
        var x;
        var x=1;
        var x="Hello World";
        console.log(x);
    </script>
</head>
<body>

</body>
</html>
```

对于上述代码，在执行的过程中首先定义了 x，并且赋值为数字 1，后将 x 再次赋值为字符串“Hello World”，然后输出。访问此页面，代码的执行并不会报错，而控制台正确打印“Hello World”，结果如图 3-40 所示。

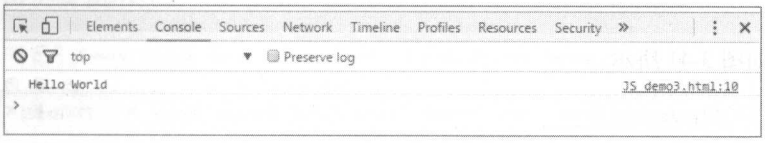


图 3-40 执行结果

1. 字符串（String）类型

字符串是存储字符的变量，值可以是引号中的任意文本、符号。引号可以是单引号或双引号，如下代码实现的效果是等效的。

```
var hello="Hello World ";
var hello='Hello World';
```

注意：可以在字符串中使用引号，只要不匹配包围字符串的引号即可。

2. 数字（Number）类型

JavaScript 只有一种数字类型，数字包含整数和浮点数部分。JavaScript 也支持通过指数的方式进行数值的定义，极大或极小的数字可以通过科学（指数）计数法来书写。

【示例 3-36】

数字类型：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
```

```
    <script>
        var x=10.00;        //使用小数点来写
        var x1=10;          //不使用小数点来写
        var x2=10e5;        //1000000
        var x3=10e-5;       //0.0001
        console.log(x);
        console.log(x1);
        console.log(x2);
        console.log(x3);
    </script>
</head>
<body>

</body>
</html>
```

显示效果如图 3-41 所示。

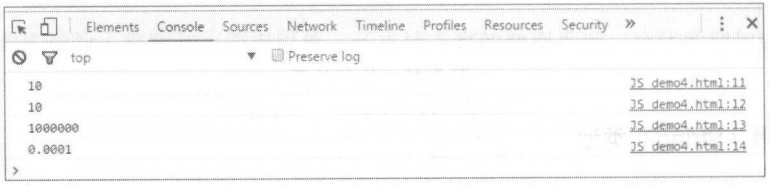


图 3-41　显示结果

3. 布尔类型

布尔（逻辑）只能有两个值：true 或 false，其一般用于逻辑判断语句。

注意：对于布尔值的赋值，false 和 true 并不能使用双引号或者单引号。否则对于 JavaScript 将会将“false”和“true”看作字符串。

【示例 3-37】

布尔类型：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var x=true;
        var y=false;
```

```
        if(x){
            console.log("这是 x 打印的");
        }else if(y){
            console.log("这是 y 打印的");
        }else{
            console.log("当 x,y 都无法打印时");
        }
    </script>
</head>
<body>

</body>
</html>
```

打印效果如图 3-42 所示。

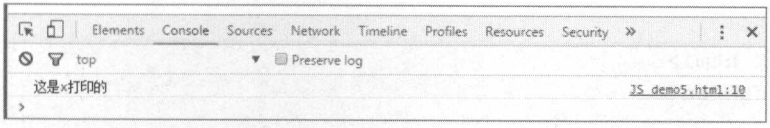


图 3-42 打印效果

4. 数组类型

数组类型是复合数据类型，用来在单独的变量名中存储一系列的值，通过下标进行取值。

【示例 3-38】

数组类型：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var x=new Array('数组 1','数组 2','数组 3','数组 4','数组 5','数组 6');
        console.log(x[3]);
    </script>
</head>
<body>

</body>
</html>
```

控制台输出结果如图 3-43 所示。上述代码输出的是数组下标为 3 的值，控制台输出结果是数组 4。

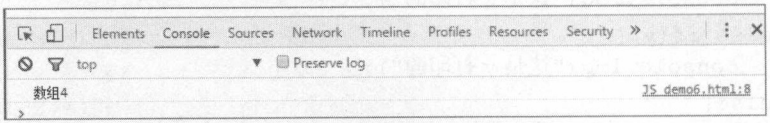


图 3-43　输出结果

注意：数组的下标是基于零的，所以数组中的第一个值的下标是[0]，第二个是[1]，依此类推。

5. 对象类型

对象是对数组的一种扩充，同样用于存储一系列相关可以构造成对象的数据集。由花括号分隔，在括号内部，对象的属性以名称和值对的形式（name:value）来定义，属性由逗号分隔。

【示例 3-39】

对象类型：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var person={name:"PersonName",sex:"PersonSex",age:"PersonAge"};
        console.log(person.name);
        console.log(person['sex']);
        console.log(person);
    </script>
</head>
<body>

</body>
</html>
```

控制台的打印显示效果如图 3-44 所示。

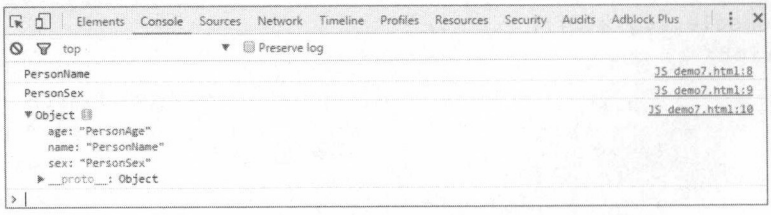


图 3-44　显示效果

对象属性有两种寻址方式，如上例代码所示。以下代码这样取值，两种方式是等效的，均可以去除 person 对象的 name 值。

```
person.name
person['name']
```

6. undefined 和 null

undefined 这个值表示变量为赋值的状态。null 特殊数据类型，表示空，对于某一变量已经定义但是不存在的值为空。

【示例 3-40】

undefined 和 null：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var x;
        var y=1;
        console.log(x);
        console.log(y);
        y=null;
        console.log(y);
    </script>
</head>
<body>

</body>
</html>
```

其显示效果如图 3-45 所示。对于定义了一个变量 x，但是未赋值，所以在控制台打印为 undefined。

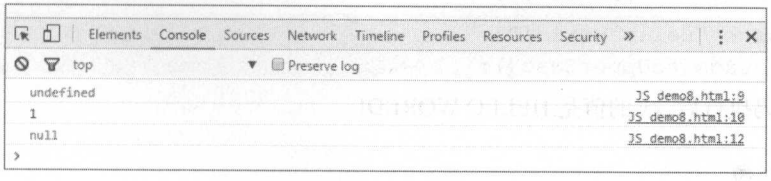


图 3-45 显示效果

注意：可以通过将变量的值设置为 null 来清空变量。

当声明新变量时，可以使用关键词 new 来声明其类型：

```
var carname=new String;
var x=new Number;
var y=new Boolean;
var cars=new Array;
var person= new Object;
```

JavaScript 中一切变量均为对象，所以当声明一个变量时，就相当于创建了一个新的 JavaScript 对象。

3.6.4 JavaScript 对象

JavaScript 中的所有事物都是对象：字符串、数字、数组、日期等。在 JavaScript 中，对象是拥有属性和方法的数据。属性是与对象相关的值，方法是能够在对象上执行的动作。JavaScript 提供多个内建对象，比如 String、Date、Array 等。JavaScript 还允许自定义对象。

1. 调用对象的属性和方法

对象只是带有属性和方法的特殊数据类型，属性是与对象相关的值。通过“.”语法形式可以访问对象的属性，如：

```
objectName.propertyName
```

JavaScript 的对象存在一些既定的属性，通过调用这些属性可以获得有关此对象的一些信息。比如 String 对象的 length 属性可以获得字符串的长度：

```
var message="Hello World!";
var x=message.length;
```

在以上代码执行后，*x* 的值将是 12。

访问对象的方法如下所示：

```
objectName.methodName()
```

使用 String 对象的 toUpperCase()方法来将字母转换为大写：

```
var message="Hello world!";
var x=message.toUpperCase();
```

在以上代码执行后，*x* 的值是 HELLO WORLD!

2. 新建对象

除了自带的对象，JavaScript 也支持用户创建新的对象。创建新对象有两种方法：

（1）定义并创建对象的实例。

使用 new Object()方法来定义新对象，然后创建新的对象实例。

【示例 3-41】

使用 new Object()方法：

```
person=new Object();
person.name="PersonName";
person.sex="PersonSex";
person.age="PersonAge";
```

这个例子创建了一个叫“person”的对象实例，并向其添加了三个属性“name”“sex”“age”。

（2）使用以下形式来创建对象实例

```
var person={name:"PersonName",sex:"PersonSex",age:"PersonAge"};
```

这里使用对象构造器进行对象的构造，需要定义一个 function 为对象构造器。

【示例 3-42】

使用对象构造器：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function person(name, sex, age) {
            this.name = name;
            this.sex = sex;
            this.age = age;
        }
        var person1=new person("person1","man","20");
        var person2=new person("person2",'woman',"18");
        console.log(person1);
        console.log(person2);
    </script>
</head>
<body>

</body>
</html>
```

通过调用 new person()方法创建新的对象实例。这样就定义了 person1 和 person 2 两个变量，这两个变量把属性添加到 person 对象中。

打印两个对象如图 3-46 所示。

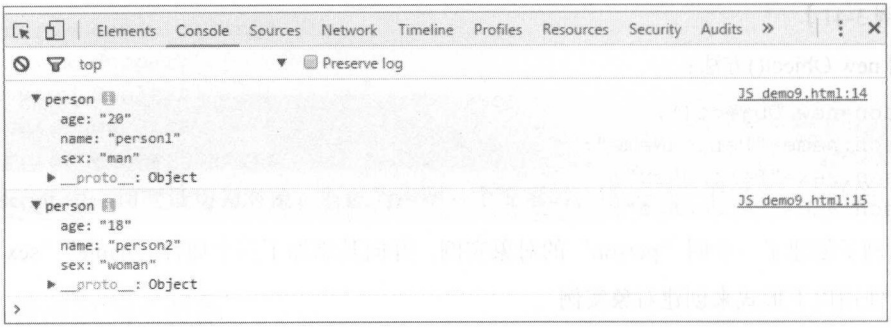


图 3-46　执行结果

当然对于已经创建的对象实例也可以更改其相应的值。

```
person.name="PersonName";
```

以上代码执行后，person.name 的值更改为“PersonName”。

3．日期对象

JavaScript 提供了日期（Date）对象，可以获得当前的日期和时间，或者对日期进行一些操作，Date 对象的常用方法如表 3-26 所示。

表 3-26　Date对象的常用方法

Date()	返回当日的日期和时间
getDate()	从 Date 对象返回一个月中的某一天(1~31)
getDay()	从 Date 对象返回一周中的某一天(0~6)
getMonth()	从 Date 对象返回月份(0~11)
getFullYear()	从 Date 对象以四位数字返回年份
getYear()	请使用 getFullYear()方法代替
getHours()	返回 Date 对象的小时(0~23)
getMinutes()	返回 Date 对象的分钟(0~59)
getSeconds()	返回 Date 对象的秒数(0~59)
getMilliseconds()	返回 Date 对象的毫秒(0~999)
getTime()	返回 1970 年 1 月 1 日至今的毫秒数
Date	对象用于处理日期和时间

【示例 3-43】

Date 对象：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var myDate=new Date();
        console.log(myDate);
        console.log(myDate.getDate());
        console.log(myDate.getDay());
        console.log(myDate.getMonth());
        console.log(myDate.getYear());
        console.log(myDate.getHours());
        console.log(myDate.getMinutes());
        console.log(myDate.getTime());
    </script>
</head>
<body>

</body>
</html>
```

其显示效果如图 3-47 所示。



图 3-47 显示效果

4. 数组对象

数组对象使用单独的变量名来存储一系列的值，可以使用下标进行值的输出。该对象也存在一些对数组操作的常用方法，如表 3-27 所示。

5. Math 对象

Math（算数）对象的作用是：执行常见的算数任务，可以进行一些数据的计算和较复杂的运算。Math 对象常用的方法如表 3-28 所示。

表 3-27　数组常用方法

方 法 名	说　明
concat()	合并多个数组
join()	用数组的元素组成字符串
pop()	删除数组的最后一个元素
push()	数组的末尾添加新的元素
reverse()	将一个数组中的元素的顺序反转排序
shift()	删除数组的第一个元素
slice()	从一个数组中选择元素
sort()	数组排序
splice()	在数组的第 2 位添加一个元素
toString()	转换数组到字符串
unshift()	在数组的开头添加新元素

表 3-28　Math对象常用方法

方法名称	说　明
round()	把数四舍五入为最接近的整数
random()	返回 0~1 之间的随机数
max(x,y,z,...,n)	返回 x,y,z,...,n 中的最高值
min(x,y,z,...,n)	返回 x,y,z,...,n 中的最低值
abs(x)	返回 x 的绝对值
acos(x)	返回 x 的反余弦值
asin(x)	返回 x 的反正弦值
atan(x)	以介于-PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值
cos(x)	返回数的余弦
floor(x)	对 x 进行下舍入
log(x)	返回数的自然对数（底为 e）

3.6.5　JavaScript 函数

函数可以理解为可重复使用的代码块，一般由事件驱动。

【示例 3-44】

使用函数：

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function myFunction()
        {
            alert("Hello World!");
        }
    </script>
</head>
<body>
<button onclick="myFunction()">单击弹出 Hello World! </button>
</body>
</html>
```

单击“测试”按钮时，会自动调用 myFunction，弹出一个弹出框，显示“Hello World”字样，如图 3-48 所示。



图 3-48 执行结果

上述代码的 myFunction()就是 JavaScript 中的一个函数，而 myFunction 则是其函数名。

1. 定义函数的相关语法

函数就是包裹在花括号中的代码块，前面使用了关键词 function 定义函数，其后为该函数的名称：

```
function functionname()
{
这里是要执行的代码
}
```

可以在某事件发生时直接调用函数（比如当用户单击按钮时），而且 JavaScript 可在任何位置进行调用。

注意：JavaScript 对大小写敏感。关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

函数名称后方小括号中存在的值或多个由逗号隔开的值，称为该函数接受的参数。调用带参数的函数时，可以向其传递值，这些值被称为参数，函数本身会接收此参数，并在函数中使用。可以

传递任意多的参数，由逗号（,）分隔，例如以下代码发送了两个参数：

```
myFunction(argument1,argument2)
```

如果需要使用到参数的传值操作，需要在声明函数时把参数作为变量来声明：

```
function myFunction(argument1, argument2)
{
这里是要执行的代码
}
```

变量和参数必须以一致的顺序出现。第一个变量就是第一个被传递参数的给定值，依此类推。

【示例 3-45】

变量的传递：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function myFunction(data)
        {
            alert(data);
        }
    </script>
</head>
<body>
<button onclick='myFunction("Hello World")'>单击弹出 Hello World! </button>
</body>
</html>
```

当按钮被单击时提示“Hello World”，上述代码并没有在函数中直接写定数据，而是通过参数的方式将数据传递至函数。结果显示如图 3-49 所示。



图 3-49　执行结果

注意：可以通过传递不同的参数来控制函数的执行结果和执行方式，可以使用不同的参数来调用该函数，这样就会给出不同的消息。

2. 函数的返回值

有的时候函数需要将其处理的数值返回至调用处，这使用 return 语句返回值。但在使用 return 语句时，函数会停止执行。也就是说，return 相当于此函数的结束标志，在 return 之后无论有什么语句将永远都不会执行。

【示例 3-46】

return 语句：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function myFunction(x, y) {
            var z = x + y;
            return z;
        }
        var data = myFunction(1, 2);
        console.log(data);
    </script>
</head>
<body>
</body>
</html>
```

上面的函数会在控制台中打印 3，执行结果如图 3-50 所示。



图 3-50 执行结果

【示例 3-47】

通过不同参数和 return 的调用，可以实现代码的不同返回值操作，使用 if 或者其他语句判断并返回合适的值。

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function myFunction(data){
            if(data==1){
                return "执行方法参数为 1";
            }else if(data==2){
                return "执行方法参数为 2";
            }
        }
        function btn_onclick(data){
            alert(myFunction(data));
        }
    </script>
</head>
<body>
<button onclick="btn_onclick(1)">参数为 1</button>
<button onclick="btn_onclick(2)">参数为 2</button>
</body>
</html>
```

这个页面中会出现两个按钮，绑定了同一个函数，但是会传递不同的参数，而函数内部又调用了 myFunction()并把参数传递给这个函数。

具体执行时，单击“参数为 1”按钮会弹出“执行方法参数为 1”，而单击“参数为 2”按钮会弹出“执行方法参数为 2”，如图 3-51 和图 3-52 所示。

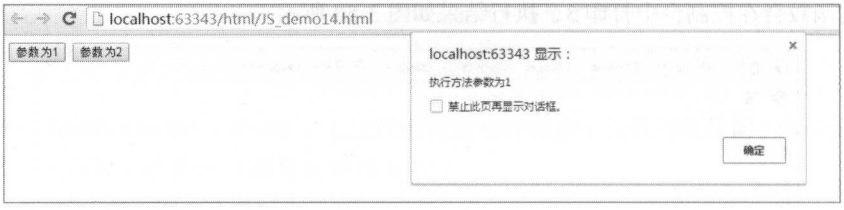


图 3-51　执行结果 1



图 3-52　执行结果 2

3.6.6 JavaScript 运算符

JavaScript 运算符主要有算术运算符、赋值运算符、比较运算符。

（1）算术运算符

算术运算符用于执行变量与/或值之间的算术运算，一些常用的算术运算符如表 3-29 所示。

表 3-29 JavaScript算术运算符

运 算 符	描 述	例 子
+	加	x=y+2
-	减	x=y-2
*	乘	x=y*2
/	除	x=y/2
%	求余数（保留整数）	x=y%2
++	累加	x=++y
--	递减	x=--y

（2）赋值运算符

赋值运算符用于给变量赋值，一些常用的赋值运算符如表 3-30 所示。

表 3-30 JavaScript赋值运算符

运 算 符	例 子	等 价 于
=	x=y	
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

（3）比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等，常用的比较运算符如表 3-31 所示。

表 3-31 JavaScript比较运算符

运 算 符	描 述	例 子
==	等于	7==8 为 false
===	全等（值和类型）	5===5 为 true；5==="5"为 false
!=	不等于	5!=8 为 true

续表

运 算 符	描　述	例　子
>	大于	5>8 为 false
<	小于	5<8 为 true
>=	大于或等于	5>=8 为 false
<=	小于或等于	5<=8 为 true

3.6.7 JavaScript 逻辑控制语句

条件语句基于不同的条件来执行不同的动作，通常用于某些代码的判断和重复执行。在写代码时，总是需要为不同的决定来执行不同的动作，此时就可以在代码中使用条件语句来完成该任务。

1. 条件判断语句

在 JavaScript 中，常用的条件判断语句有 4 种。

- if 语句：只有当指定条件为 true 时，才使用该语句来执行代码。
- if...else 语句：当条件为 true 时执行代码，当条件为 false 时执行其他代码。
- if...elseif....else 语句：使用该语句来选择多个代码块之一来执行。
- switch 语句：使用该语句来选择多个代码块之一来执行。

2. if 语句

只有当指定条件为 true 时，该语句才会执行代码。if 语法：

```
if (条件)
  {
  只有当条件为 true 时才执行代码
  }
```

注意：必需使用小写 if，使用大写字母 IF 会出现错误。

【示例 3-48】

当输入一个大于 10 的数字时，会执行在 if 之后包含的代码。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function myFunction(){
```

```
            var data=document.getElementById('data').value;
            if(data>10){
                alert("输入数字大于10")
            }
        }
    </script>
</head>
<body>
<label>和10进行数值对比</label>
<input type="text" id="data"value="">
<button onclick="myFunction()">获取输入的值</button>
</body>
</html>
```

运行时如果输入任意小于 10 的数字时不会出现任何变化，但如果输入数字大于 10 则会弹出提示。具体执行结果如图 3-53 所示。

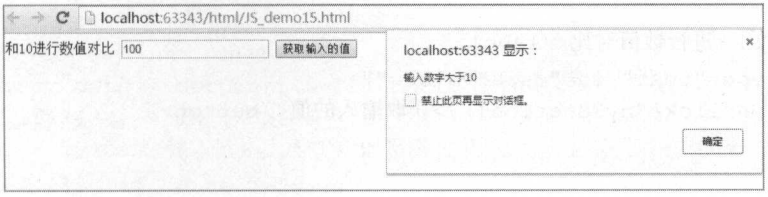


图 3-53 执行结果

注意：在这个语法中，没有...else...。已经告诉浏览器只有在 if 指定条件为 true 时，才执行代码。

3. if...else 语句

可以使用 if....else 语句在条件为 true 时执行代码，当 if 的判断条件为 false 时将会执行 else 之后的代码。if....else 语法：

```
if (条件)
  {
  当条件为 true 时执行的代码
  }
else
  {
  当条件不为 true 时执行的代码
  }
```

【示例 3-49】

将上例进行更改，输入小于 10 的数字后单击按钮会弹出提示框。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function myFunction(){
            var data=document.getElementById('data').value;
            if(data>10){
                alert("输入数字大于 10")
            }else{
                alert("输入数字小于 10")
            }
        }
    </script>
</head>
<body>
<label>和 10 进行数值对比</label>
<input type="text" id="data"value="">
<button onclick="myFunction()">获取输入的值</button>
</body>
</html>
```

当执行此代码时，输入小于 10 的数字后，将会弹出提示，具体执行效果如图 3-54 所示。

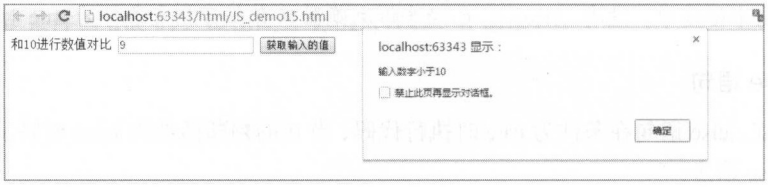


图 3-54　执行结果

4．if...else if...else 语句

使用 if....else if...else 语句来选择多个代码块之一来执行，代码执行时先判断 if 条件，如果条件不正确，返回 false 后会判断第一个 else if 的条件，如果不成功，还会接着判断下一个 else if 的条件，如果判断成功，返回 true，如果直到 else 一直没有返回 true，则会直接执行 else。if...else if...else 语法：

```
if (条件 1)
  {
  当条件 1 为 true 时执行的代码
  }
```

```
else if (条件2)
  {
  当条件2为 true 时执行的代码
  }
else
  {
  当条件1和条件2都不为 true 时执行的代码
  }
```

【示例 3-50】

依旧对上例进行改造，增加一个 else if，对比是否超过 10 但不超过 100 的情况。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function myFunction() {
            var data = document.getElementById('data').value;
            if (data > 10 && data < 20) {
                alert("输入数字大于10,但是不大于20")
            } else if (data > 20) {
                alert("输入数字大于20")
            } else {
                alert("输入数字小于10")
            }
        }
    </script>
</head>
<body>
<label>和10进行数值对比</label>
<input type="text" id="data" value="">
<button onclick="myFunction()">获取输入的值</button>
</body>
</html>
```

当输入数值在 10~20 的范围内，执行 if 之后的代码，会弹出提示“输入数字大于 10，但是不大于 20”，当输入值大于 20 时，执行 else if 之后的代码，弹出“输入数字大于 20”的提示。具体执行结果如图 3-55 和图 3-56 所示。

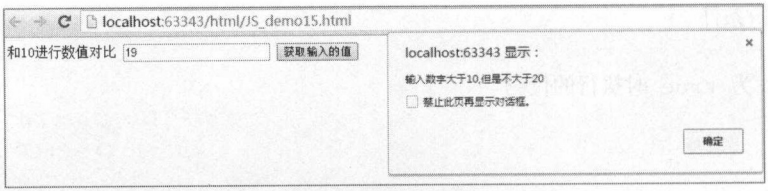


图 3-55　执行结果 1

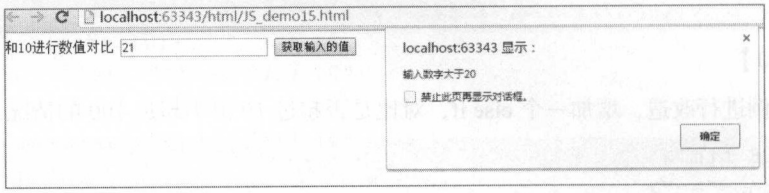


图 3-56　执行结果 2

5. switch 语句

可以使用 switch 语句来选择要执行的多个代码块之一，用于多个逻辑判断的内容，对比 case 的值，进行代码块的执行。switch 语法：

```
switch(n)
{
case 1:
  执行代码块 1
  break;
case 2:
  执行代码块 2
  break;
default:
  n 与 case 1 和 case 2 不同时执行的代码
}
```

工作原理：首先设置表达式 *n*（通常是一个变量），随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行，此时使用 break 来阻止代码自动地向下一个 case 运行。

【示例 3-51】

本例会在控制台打印显示今日的周名称。使用 Date 对象进行当前日期的获取，使用 getDay() 函数进行星期的获取。函数的返回值为 0~6 这 7 个数字，请注意 Sunday=0， Monday=1，Tuesday=2……

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var day=new Date().getDay();
        switch (day)
        {
            case 0:
                x="Today it's Sunday";
                break;
            case 1:
                x="Today it's Monday";
                break;
            case 2:
                x="Today it's Tuesday";
                break;
            case 3:
                x="Today it's Wednesday";
                break;
            case 4:
                x="Today it's Thursday";
                break;
            case 5:
                x="Today it's Friday";
                break;
            case 6:
                x="Today it's Saturday";
                break;
        }
        console.log(x);
    </script>
</head>
<body>
</body>
</html>
```

打开此页面时，会自动在控制台中打印当前的时间对应的星期数，其显示结果如图 3-57 所示。

注意：对于 switch 语句而言，使用 break 语句在每一个条件后是很有必要的一件事，否则将会自动向下继续执行。

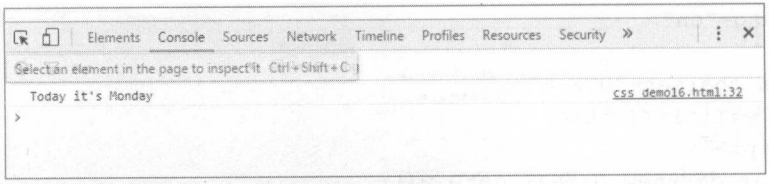


图 3-57　显示结果

6. default 关键词

可以使用 default 关键词来规定匹配不存在时做的事情，这个关键词的意义在对逻辑判断增加一个默认的选项。

在 switch 中使用 default 来确保经过 switch 条件筛选的值都经过了处理。比如如下示例将对显示星期的例子进行更改，合并周六和周日为周末。

【示例 3-52】

如果今天不是周六或周日，则输出默认的消息。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var day=new Date().getDay();
        switch (day)
        {
            case 1:
                x="Today it's Monday";
                break;
            case 2:
                x="Today it's Tuesday";
                break;
            case 3:
                x="Today it's Wednesday";
                break;
            case 4:
                x="Today it's Thursday";
                break;
            case 5:
                x="Today it's Friday";
                break;
```

```
            default:
                x="Today it's Weekend";
        }
        console.log(x);
    </script>
</head>
<body>
</body>
</html>
```

可以通过更改系统时间的方式将时间改为周六或者周日，而返回值 0 和 6 这两种情况是没有对应的 case 值的，所以会执行 default 之后的代码，显示结果如图 3-58 所示。

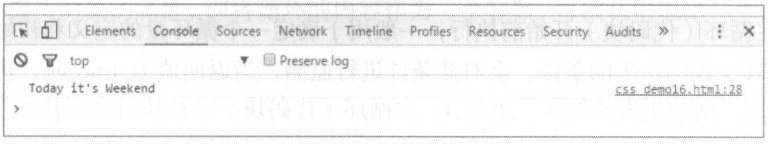


图 3-58 显示结果

7. 循环语句

JavaScript 提供了几种循环语句可以循环代码块执行指定的次数，通过判断条件是否成立来确定下次是否再次执行。

如果需要输出一个数组的全部元素，不用循环语句进行输出的话，则需手动多次输出。

```
alert(array[0]);
alert(array[1]);
alert(array[2]);
alert(array[3]);
……
```

对于只存在几个元素的数组而言，是可以使用上例这样的代码进行输出的，但如果是很多元素的一个数组，这样重复的代码便变得烦琐且冗余。这个时候，可以使用循环语句进行输出：

```
for (var i=0;i<array.length;i++)
{
alert(array[i]);
}
```

上述代码与前面实现的效果是相同的。

JavaScript 为开发者提供了不同类型的循环语句。JavaScript 支持 4 种类型的循环。

- for：循环代码块一定的次数。

- for/in：循环遍历对象的属性。
- while：当指定的条件为 true 时循环指定的代码块。
- do/while：同样当指定的条件为 true 时循环指定的代码块。

8. for 循环

for 循环是创建循环时常会用到的语句。for 语法：

```
for (语句 1; 语句 2; 语句 3)
  {
  被执行的代码块
  }
```

语句 1：在循环（代码块）开始前执行，一般用于确定一个循环量的定义和初始化赋值。语句 2：定义运行循环（代码块）的条件，会对此条件进行监测，当返回值为 true 时，执行循环体；当返回值为 false 时，则会跳出该循环。语句 3：在循环（代码块）已被执行之后执行。

【示例 3-53】

for 循环：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        var data='';
        for(i=1;i<10;i++){
            for(j=1;j<=i;j++){
                data=data+i+"*"+j+"="+(i*j)+" "
            }
            data=data+"\n";
        }
        console.log(data)
    </script>
</head>
<body>

</body>
</html>
```

上例可以在控制台中打印一个 9×9 乘法口诀表，显示效果如图 3-59 所示。

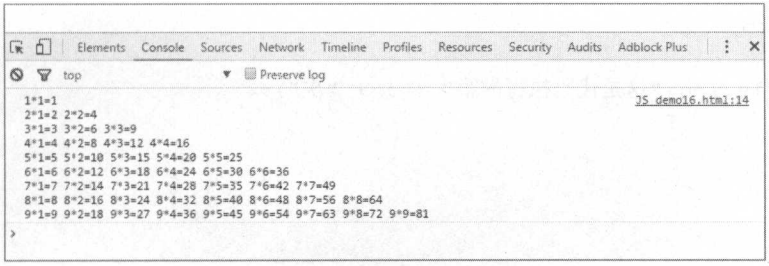


图 3-59 显示效果

9. for/in 循环

for/in 循环遍历对象的属性，即将循环的内容作为一个容器，每次从中取出一个需要的项目进行输出。

【示例 3-54】

for/in 循环：

```
var person={fname:"John",lname:"Doe",age:25};

for (x in person)
  {
  txt=txt + person[x];
  }
```

10. while 循环

while 循环会在指定条件为真时循环执行代码块，在每次执行完成代码块之后进行条件的判断，如果为 true 则再次执行代码。while 语法：

```
while (条件)
  {
  需要执行的代码
  }
```

注意：只要指定条件为 true，循环就可以一直执行代码。

【示例 3-55】

while 循环，本例中的循环将持续运行，只要变量 i 小于 5。

```
<html>
<head>
<script>
var i=1;
```

```
while (i<5)
{
    console.log("这是代码执行的第" + i + "次");
    i++;
}
</script>
</head>
<body>
</body>
</html>
```

当 i 小于 5 的时候执行代码，在控制台中打印输出此代码执行的次数。输出结果如图 3-60 所示。

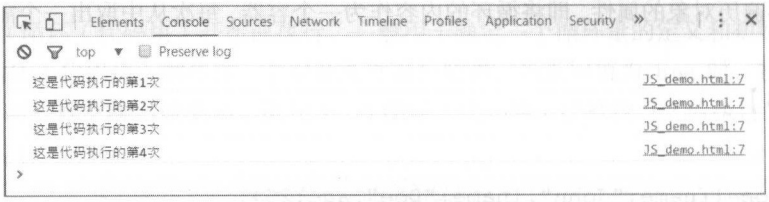


图 3-60　输出结果

注意：如果忘记增加条件中所用变量的值，该循环永远不会结束，可能导致浏览器崩溃。

11. do/while 循环

do/while 循环是 while 循环的变体。在检查条件是否为真之前，该循环会执行一次代码块，如果条件为真，就会重复这个循环。do/while 循环的语法：

```
do
  {
  需要执行的代码
  }
while (条件);
```

【示例 3-56】

do/while 循环至少会执行一次，即使条件是 false，隐藏代码块也会在条件被测试前执行。

```
<html>
<head>
<script>
var i=1;
do
{
    console.log("这是代码执行的第" + i + "次");
```

```
    i++;
}
while (i<1);
</script>
</head>
<body>
</body>
</html>
```

do/while 循环与 while 循环的差别在于，使用 do/while 循环时会无条件先执行一次 do 中的代码段，再去判定条件是否成立，返回 true 则执行第二次，如果为 false 则中断。

所以对于上述实例中的代码，依旧会在控制台中输出一次执行结果，结果如图 3-61 所示。

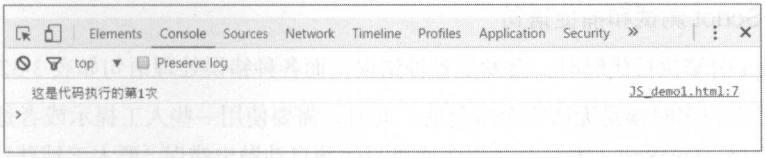


图 3-61 执行结果

12. continue 语句

continue 语句中断循环中的迭代，如果出现了指定的条件，则继续循环中的下一个迭代。

【示例 3-57】

continue 语句：

```
<html>
<head>
<script>
var i=1;
while (i<5)
{
    if(i==2){
        i++;
        continue;
    }
    console.log("这是代码执行的第" + i + "次");
    i++;
}
</script>
</head>
<body>
```

```
</body>
</html>
```

当循环时，i 的值为 2 则执行 continue 操作，直接跳出此次循环进入下次循环。执行效果如图 3-62 所示。

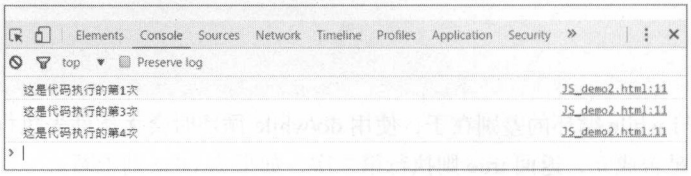


图 3-62　执行效果

13. JavaScript 测试和捕捉语句

当 JavaScript 引擎执行代码时，会发生各种错误，而各种错误处理语句如表 3-32 所示。

代码中的错误很多时候是无法完全避免的，此时，需要使用一些人工提示或者是中断程序，并且捕获当前的错误。JavaScript 本身会对语法或拼写问题自动抛出错误，要人工捕获这些错误，需要用到 try…catch 语句。

表 3-32　错误处理语句

语　句	说　明
try	语句测试代码块的错误
catch	语句处理错误
throw	语句创建自定义错误

- try 语句定义在执行时进行错误测试的代码块。
- catch 语句定义当 try 代码块发生错误时，所执行的代码块。

try 和 catch 是成对出现的，语法：

```
try
  {
  //在这里运行代码
  }
catch(err)
  {
  //在这里处理错误
  }
```

【示例 3-58】

在 while 循环例子中故意在代码块的代码中不定义变量 i，同时使用 try…catch 语句进行包裹。

catch 块会捕捉到 try 块中的错误，并执行代码来处理它。

```
<html>
<head>
<script>
try{
while (i<5)
{
    if(i==2){
        i++;
        continue;
    }
    console.log("这是代码执行的第" + i + "次");
    i++;
}
}
catch(err){
    console.log(err)
}
</script>
</head>
<body>
</body>
</html>
```

如果不使用 try 来捕获此错误，系统会自动报错并且中断程序，在控制台会以红色的信息进行输出，而使用 try...catch 语句捕获后，使用 log()打印则会以普通信息的形式进行控制台输出，显示效果如图 3-63 所示。

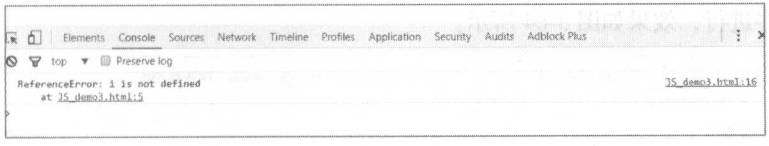


图 3-63 显示效果

14. throw 语句

throw 语句允许开发者创建自定义错误，正确的技术术语是：创建或抛出异常（exception）。如果把 throw 语句与 try 和 catch 一起使用，那就能够控制程序流，并生成自定义的错误消息。throw 语句：

```
throw exception
异常可以是 JavaScript 字符串、数字、逻辑值或对象。
```

【示例 3-59】

本例检测输入变量的值。如果值是错误的，会抛出一个异常（错误）。catch 会捕捉到这个错误，并显示一段自定义的错误消息。

```
<html>
<head>
<script>
try{
var i=1
while (i<5)
{
    if(i==2){
       throw "执行到 2 发生错误"
    }
    console.log("这是代码执行的第" + i + "次");
    i++;
}
}
catch(err){
    console.log(err)
}
</script>
</head>
<body>
</body>
</html>
```

此实例依旧是更改 while 循环的实例，通过对于执行到 i==2 时，手动使用 throw 语句来抛出一个错误中断程序进行，效果如图 3-64 所示。

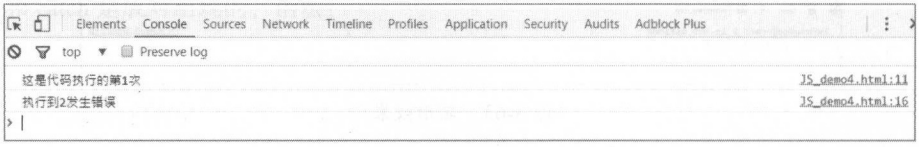


图 3-64　打印效果

3.7　小结

通过对 HTML、CSS、JavaScript 的学习，开发者可以开发出适合应用场景的页面和业务逻辑。本书并非一本介绍网页开发的书籍，只是由于在微信开发小程序的基础上使用到了与之相关的技术

基础，所以本章简略而快速地讲解了这些基础。

通过 HTML 的学习，读者可以认知到不同标签的使用和其意义。

通过 CSS 的学习，读者可以针对 HTML 中的标签样式进行定义和更新。

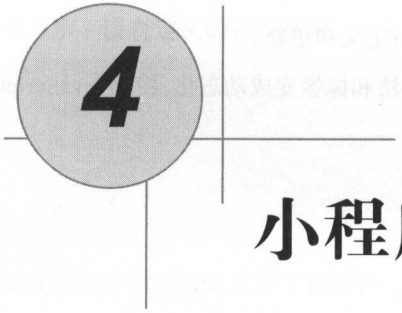
通过 JavaScript 的学习，更是将只用 HTML 和 CSS 的静态语法和标签变成动态化，使用 JavaScript 逻辑代码的编写，完成所需要的页面逻辑。

3.8 实践与练习

对于代码的熟悉程度，是离不开对于代码的编辑和练习的。所以对于本章知识点的学习最好的是进行不断练习和对于实际需求的编码。

- 对于本章的所有实例进行自己的尝试和开发。
- 除了基础的 HTML 页面、CSS、JavaScript 单一实例的编写以外，可以通过将三者进行结合写出更加美观的页面。
- 结合下一章的内容，进行三种技术的对比，打好小程序的基础。

请读者自行进行 HTML 页面的编写，根据不同的网站进行模仿设计和进行具体的编码任务。



小程序开发组件

微信小程序的应用框架为开发者们提供了一系列使用简单的基础组件，每一个开发者都可以通过对于这些基础组件的组合和属性的调整快速开发小程序，并且能实现一些不错的效果。

本章主要涉及的知识点有：

- 基本的微信小程序组件的介绍和示例。
- 学会综合不同的属性和组件，形成自己的小程序。
- 通过本章的技术和示例，演示不同的组件使用规范，制作出不同的显示效果，做出自己的小程序界面。

注意：本章内容不包含后台 JS 逻辑。

4.1 视图组件

什么是微信小程序的组件：

- 组件是视图层的基本组成单元。
- 组件自带一些功能与微信风格的样式。
- 一个组件通常包括开始标签和结束标签，属性用来修饰这个组件，内容在两个标签之内。

4.1.1 视图容器（view）组件

微信小程序的组件通常包括开始标签和结束标签，写法与 XML 或闭合的 HTML 类似。view 组件代表了一个页面的基本视图，也就是一个新的页面的最外层容器，相当于 HTML 中的 DIV 容器。本节用一个例子来说明视图容器。

【示例 4-1】

新建一个微信小程序项目：

（1）打开微信开发 IDE。如果是第一次打开，直接可以新建项目；如果之前已经有一个项目，IDE 会自动打开之前的项目，这样，就需要先单击“项目选项”按钮，然后单击左下角的“关闭”按钮，最后在新的页面里单击“添加项目”按钮。

（2）新项目的配置如图 4-1 所示，单击“添加项目”按钮。

（3）此时可以看到一个 Hello World 程序，其中的图片是登录微信账户的头像，如图 4-2 所示，而这个界面所显示的图像，就是一个 view 组件。

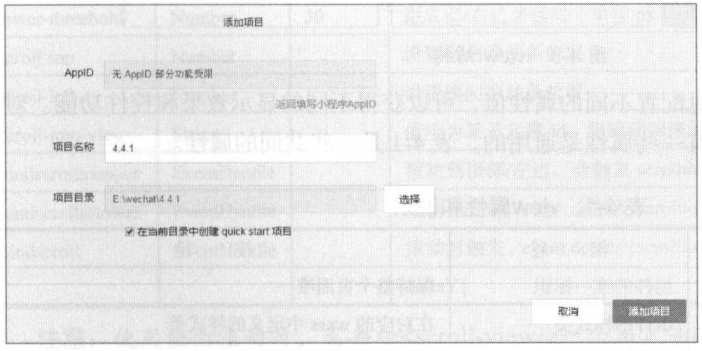


图 4-1 添加新项目

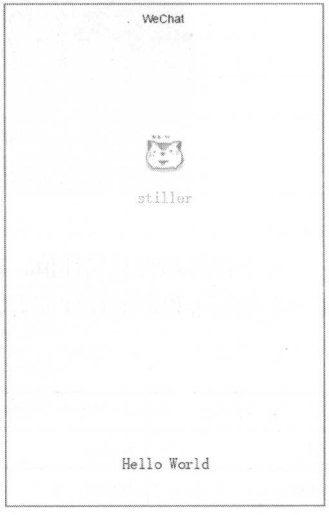


图 4-2 view 标签示例

（4）选择编辑界面，打开 pages->index->index.wxml 可以看到具体代码：

```
<view class="container">
    <view  bindtap="bindViewTap" class="userinfo">
        <image ……></image>
        <text class="userinfo-nickname">{{userInfo.nickName}}</text>
    </view>
    <view class="usermotto">
        <text class="user-motto">{{motto}}</text>
    </view>
</view>
```

【代码解析】

上述的代码和 HTML 或 XML 代码是一样的结构。存在三个<view></view>控件，两个 view 嵌套于一个 view 中，形成了两层的样式，具体结构如图 4-3 所示。可以看出对于整个 view 容器覆盖了整个 page 的显示，包含了第二个、第三个 view。其具体的样式是由该文件夹下同名的.wxss 控制的。

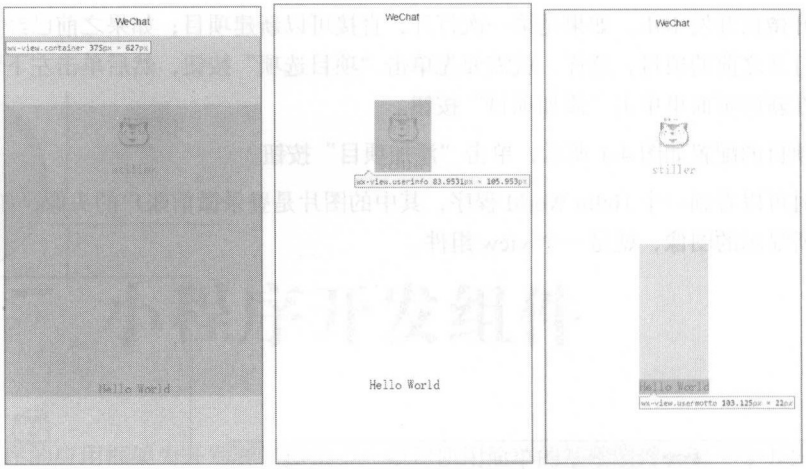


图 4-3　view 结构

每个控件都有其属性值，通过配置不同的属性值，可以获得不同的显示效果和控件功能。对于每一个微信小程序的控件而言，有一些属性是通用的。表 4-1 是一些共同的属性。

表 4-1　view属性和说明

属 性 名	类　　型	描　　述	注　　释
Id	String	组件的唯一标识	保持整个页面唯一
class	String	组件的样式类	在对应的 wxss 中定义的样式类
style	String	组件的内联样式	可以动态设置的内联样式
hidden	Boolean	组件是否显示	所有组件默认显示
data-*	Any	自定义属性	组件上触发事件时，会发送给事件处理函数
bind*/catch*	EventHandler	组件的事件	不同的事件处理

表 4-2 是每个属性的类型。

表 4-2　属性类型

类　　型	描　　述	注　　解
Boolean	布尔值	组件写上该属性值为 true，不写为 false
Number	数字	例如：1、2.5
String	字符串	例如："string"
Array	数组	例如：[1, "string"]
Object	对象	例如：{ key: value }
EventHandler	处理函数名	Page 中定义的事件处理函数名
Any	任意属性	

注意：所有组件与属性都要小写，以连字符“-”连接，本节代码中没有列出<image>标签中的属性。

4.1.2 可滚动视图区域（scroll-view）

对于可滚动视图区域，其属性和类型默认值已经说明，如表 4-3 所示。

表 4-3 scroll-view属性

属 性 名	类 型	默认值	说 明
scroll-x	Boolean	false	允许横向滚动
scroll-y	Boolean	false	允许纵向滚动
upper-threshold	Number	50	距顶部/左边多远时（单位 px），触发 scrolltoupper 事件
lower-threshold	Number	50	距底部/右边多远时（单位 px），触发 scrolltolower 事件
scroll-top	Number		设置竖向滚动条位置
scroll-left	Number		设置横向滚动条位置
scroll-into-view	String		值应为某子元素 id，则滚动到该元素，元素顶部对齐滚动区域顶部
bindscrolltoupper	EventHandle		滚动到顶部/左边，会触发 scrolltoupper 事件
bindscrolltolower	EventHandle		滚动到底部/右边，会触发 scrolltolower 事件
bindscroll	EventHandle		滚动时触发，event.detail={scrollLeft,scrollTop,scrollHeight,scrollWidth,deltaX,deltaY}

注意：使用竖向滚动时，需要给<scroll-view/>一个固定高度，通过 WXSS 设置 height。

4.1.3 滑动界面（swiper）

滑动界面区域的具体属性和类型说明如表 4-4 所示。

表 4-4 swiper的属性和类型说明

属 性 名	类 型	默 认 值	说 明
indicator-dots	Boolean	false	是否显示界面指示点
autoplay	Boolean	false	是否自动切换
current	Number	0	当前所在页面的 index
interval	Number	5000	自动切换时间间隔
duration	Number	1000	滑动动画时长
bindchange	EventHandle		current 改变时会触发 change 事件

【示例 4-2】

一个滑动界面的例子：

```
<swiper indicator-dots="{{indicatorDots}}"
  autoplay="{{autoplay}}" interval="{{interval}}" duration= "{{duration}}">
  <block wx:for="{{imgUrls}}">
    <swiper-item>
      <image src="{{item}}" class="slide-image" width="355" height= "150"/>
    </swiper-item>
  </block>
</swiper>
<button bindtap="changeIndicatorDots"> indicator-dots </button>
<button bindtap="changeAutoplay"> autoplay </button>
<slider bindchange="intervalChange" show-value min="500" max="2000"/>
interval
<slider bindchange="durationChange" show-value min="1000" max="10000"/>
duration
```

页面的代码：

```
Page({
  data: {
    imgUrls: [
      'http://img02.tooopen.com/images/20150928/tooopen_sy_
143912755726.jpg',
      'http://img06.tooopen.com/images/20160818/tooopen_sy_
175866434296.jpg',
      'http://img06.tooopen.com/images/20160818/tooopen_sy_
175833047715.jpg'
    ],
    indicatorDots: false,
    autoplay: false,
    interval: 5000,
    duration: 1000
  },
  changeIndicatorDots: function(e) {
    this.setData({
      indicatorDots: !this.data.indicatorDots
    })
  },
  changeAutoplay: function(e) {
    this.setData({
      autoplay: !this.data.autoplay
    })
  },
  intervalChange: function(e) {
    this.setData({
      interval: e.detail.value
    })
  },
```

```
  durationChange: function(e) {
    this.setData({
      duration: e.detail.value
    })
  }
})
```

注意：其中只可放置<swiper-item/>组件，其他节点会被自动删除，仅可放置在<swiper/>组件中，宽高自动设置为 100%。

本例效果如图 4-4 所示。

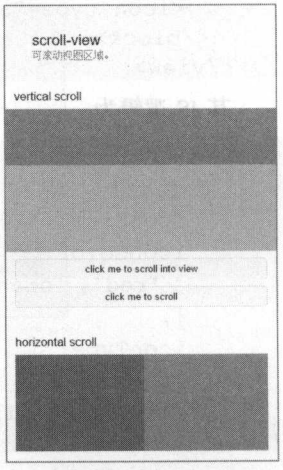


图 4-4 滑动界面示例

4.2 基础内容

微信小程序中提供了一些 view 中的小组件，作为基础布局中的基本套件内容，其中包含了图标、文本、进度条等内容。本节就来介绍这些小组件。

4.2.1 小图标（icon）

小图标的属性和类型说明如表 4-5 所示。

表 4-5 icon的属性和说明

属性名	类型	默认值	说明
type	String		icon 的类型，有效值：success、success_no_circle、info、warn、waiting、cancel、download、search、clear
size	Number	23	icon 的大小，单位 px
color	Color		icon 的颜色，同 CSS 的 color

【示例 4-3】

小图标：

```
<view class="group">
  <block wx:for="{{iconSize}}">
    <icon type="success" size="{{item}}"/>
  </block>
</view>

<view class="group">
  <block wx:for="{{iconType}}">
    <icon type="{{item}}" size="45"/>
```

```
    </block>
  </view>


  <view class="group">
    <block wx:for="{{iconColor}}">
      <icon type="success" size="45" color="{{item}}"/>
    </block>
  </view>
```

其 JS 逻辑为：

```
Page({
  data: {
    iconSize: [20, 30, 40, 50, 60, 70],
    iconColor: [
      'red', 'orange', 'yellow', 'green', 'rgb(0,255,255)', 'blue', 'purple'
    ],
    iconType: [
      'success', 'info', 'warn', 'waiting', 'safe_success', 'safe_warn',
      'success_circle', 'success_no_circle', 'waiting_circle', 'circle',
'download',
      'info_circle', 'cancel', 'search', 'clear'
    ]
  }
})
```

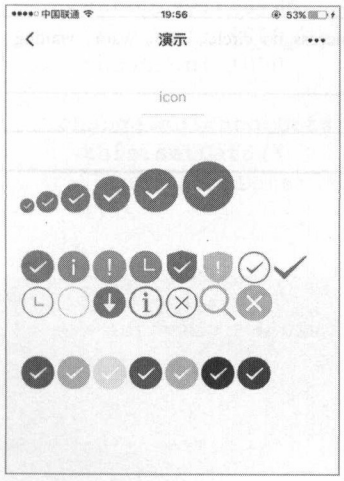


图 4-5　icon 显示效果

对于代码的样式和颜色都可以通过 page 属性进行控制和调整，效果如图 4-5 所示。

4.2.2　文本组件（text）

文本组件用于展示文字内容与说明内容，它和其他所包含文字的组件有个非常大的不同，就是其内容长按时是可以被选中的。选中的内容可以复制或执行搜索等操作，正如普通网站中 标签中的文字一样。

注意：<text></text>这对标签包含的文字支持‘\’转义符，并且其中的文字是默认自动换行的。

【示例 4-4】

文本组件，新建一个项目，如图 4-6 所示。

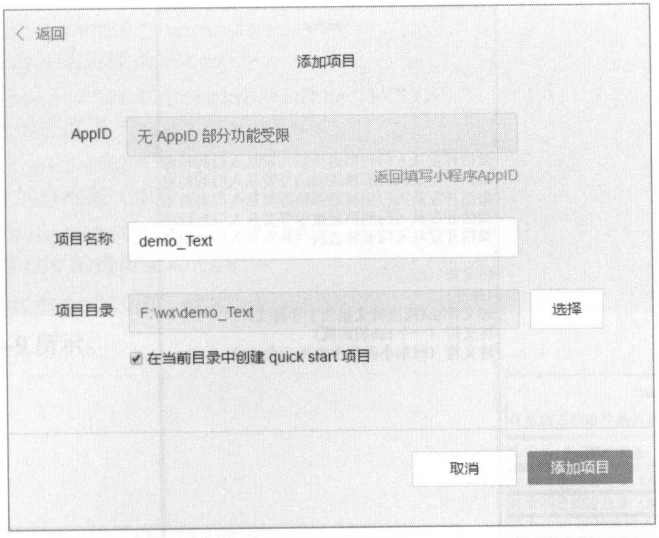


图 4-6 新建项目

单击“添加项目”选项，在主页 index.wxml 中编写代码：

```
<!--index.wxml-->
<!--index.wxml-->
<view class="container">
  <view class="body-view">
    <text>
      微信开发从入门到精通微信开发从入门到精通
      微信开发从入门到精通微信开发从入门到精通
      微信开发从入门到精通微信开发从入门到精通
      微信开发从入门到精通微信开发从入门到精通
      微信开发从入门到精通微信开发从入门到精通
    </text>
    <text>
      转义符\n(换行)
      转义符\\n(两次转义相当于不转义)
      转义符\t(一个 tab 的距离)
      转义符\r(回车小程序中也为一个 tab)
    </text>
  </view>
</view>
```

按 Ctrl+s 组合键保存后，查看显示效果如图 4-7 所示。

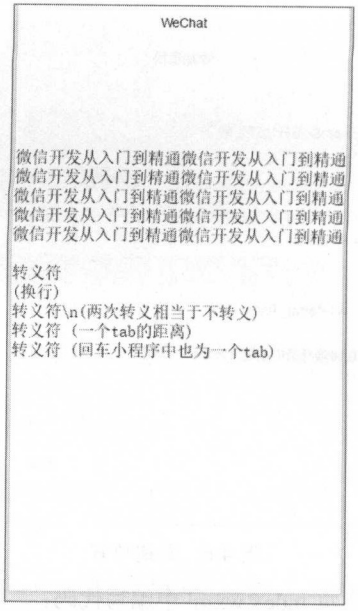


图 4-7 text 显示效果

4.2.3 进度条组件（progress）

进度条组件用于某一操作上的进度处理，主要是提供良好的用户体验，通过实时更改其显示效果，可以实现用户对于某一种操作或者是载入等待时间的把控，使整个过程更具有互动和合理性。进度条组件的属性如表 4-6 所示。

表 4-6 progress属性

属 性 名	类 型	默 认 值	说 明
percent	Float	无	百分比 0~100
show-info	Boolean	false	在进度条右侧显示百分比
stroke-width	Number	6	进度条线的宽度，单位 px
color	Color	#09BB07	进度条颜色
active	Boolean	false	进度条从左往右的动画

【示例 4-5】

进度条组件：新建项目 demo_progress，如图 4-8 所示。

单击编辑选项卡，打开编辑窗口，再打开位于 psges/index/index.wxml 中的文件，编辑代码：

```
<text>任意状态的进度条并且在右侧显示百分百</text>
```

```
<progress percent="20" show-info />
<text>宽度为 12px 的进度条</text>
<progress percent="20" stroke-width="12"/>
<text>任意颜色的进度条（red）</text>
<progress percent="60" color="red" />
<text>任意颜色的进度条（RGB 值）</text>
<progress percent="60" color="#000" />
<text>动态达到 100 的进度条</text>
<progress percent="100" active />
```

显示效果如图 4-9 所示。

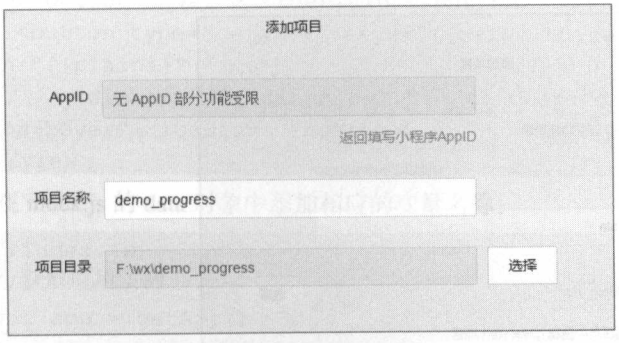


图 4-8 新建项目

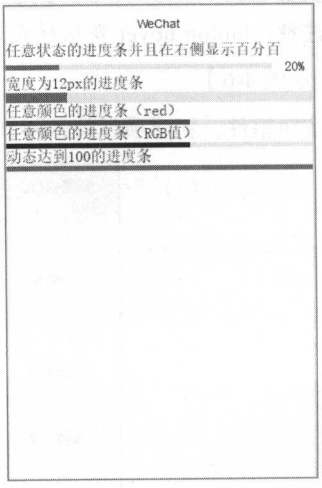


图 4-9 进度条显示效果

注意：进度条支持的颜色类型是和网页 CSS 相似的，除了基本的 RGB 类型，还支持常用的英文种类的颜色，比如 red、green、blue 等。

4.3 表单组件

表单是网页开发中非常重要的一部分，对于类似的微信小程序而言，表单技术同样在小程序的开发中非常重要。本节就来学习表单。

4.3.1 按钮组件（button）

微信小程序中已经帮助每一位开发者集成了统一风格的 button 组件，对于开发者而言，只要通过参数的微调，就可以制作出属于自己的按钮组件。按钮的属性如表 4-7 所示。

表 4-7　button的属性

属 性 名	类　型	默 认 值	说　明
Size	String	default	按钮的大小，有效值 default、mini
Type	String	default	按钮的样式，有效值 primary、default、warn
Plain	Boolean	false	按钮是否镂空，背景色透明
Disable	Boolean	false	是否禁用
Loading	Boolean	false	名称前是否带 loading 图标
Form-type	String	无	有效值：submit、reset 用于<form/>组件，单击分别会触发 submit/reset 事件
Hover-class	String	Button-hover	指定按钮按下去的样式类，当 hover-class="none"时，没有单击效果

注意：button-hover 默认样式为{background-color:rgba(0,0,0,0.1);opacity:0.7;} 。

【示例 4-6】

按钮组件，新建项目如图 4-10 所示。

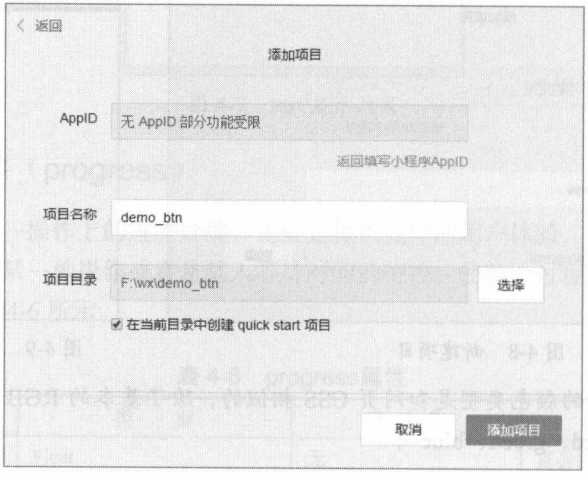


图 4-10　新建项目

编写 index.wxml 的页面代码：

```
<!--index.wxml-->
<view>
  <button type="default" size="{{defaultSize}}" loading= "{{loading}}"
plain="{{plain}}"
      disabled="{{disabled}}" bindtap="default" hover-class= "other-
button-hover"> default </button>
  <button type="primary" size="{{primarySize}}" loading= "{{loading}}"
plain="{{plain}}"
```

```
        disabled="{{disabled}}" bindtap="primary"> primary </button>
    <button type="warn" size="{{warnSize}}" loading="{{loading}}" plain=
"{{plain}}"
        disabled="{{disabled}}" bindtap="warn"> warn </button>
    <button type="default" size="{{miniSize}}" loading="{{loading}}" plain=
"{{plain}}"
        disabled="{{disabled}}"                    bindtap="default"
hover-class="other-button-hover"> minidefault </button>
    <button type="default" size="{{defaultSize}}" loading="{{loading}}"
plain="{{plain}}"
        disabled="{{trueDisabled}}"    bindtap="default"    hover-class=
"other-button-hover"> Disable </button>
    <button type="default" size="{{defaultSize}}" loading="{{loading}}"
plain="{{truePlain}}"
        disabled="{{disabled}}"      bindtap="default"      hover-class=
"other-button-hover"> truePlain </button>
    <button type="default" size="{{defaultSize}}" loading="{{trueLoading}}"
plain="{{plain}}"
        disabled="{{disabled}}"  bindtap="default"  hover-class="other-
button-hover"> loading </button>
  </view>
```

在 index.js 的 data 对象中添加相应的变量名称：

```
//index.js
//获取应用实例
var app = getApp()
Page({
  data: {
    defaultSize: 'default',
    primarySize: 'default',
    warnSize: 'default',
    disabled: false,
    plain: false,
    loading: false,
    miniSize:'mini',
    truePlain:true,
    trueDisabled:true,
    trueLoading:true,
  },
  //事件处理函数
  bindViewTap: function() {
    wx.navigateTo({
```

```
      url: '../logs/logs'
    })
  },
  onLoad: function () {
    console.log('onLoad')
    var that = this
    //调用应用实例的方法获取全局数据
    app.getUserInfo(function(userInfo){
      //更新数据
      that.setData({
        userInfo:userInfo
      })
    })
  }
})
```

实现效果如图 4-11 所示。

- 第一个按钮是 default 样式，也就是什么都不设置时的按钮样式。
- 第二个按钮是 primary 样式，这个样式适合于对于一些页面的主要操作按钮，可以起到相应的提醒功能。
- 第三个按钮是 warn 样式，是按钮的警告按钮，红色的醒目颜色。
- 第四个按钮是把 default 按钮的 size 设置成 mini 时的样式大小。
- 第五个按钮是把 default 按钮设置成不可操作时的样式，即对于 disabled 样式设置成 true。
- 第六个按钮是把 default 样式的 Plain 属性设置成 true 时的样式。
- 第七个按钮是把 default 样式的按钮 loading 属性设置成 true 时的样式，此时按钮会在基本的样式前面加载一个载入的动画，可以在等待跳转或者是处理数据的情况下使用此样式。

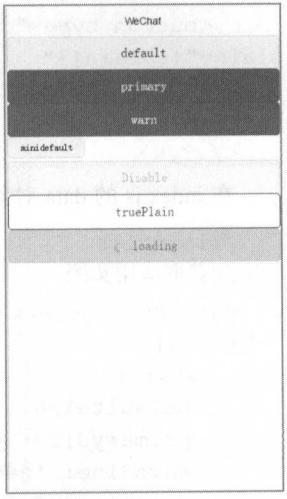


图 4-11　按钮显示示例

4.3.2　多选择框（checkbox）

微信小程序的开发也为开发者们提供了统一样式的多选框，用于一些表单的多项勾选内容或是设置时的多选菜单等。相对于样式多样的按钮组件而言，多选框的样式选择比较单一。使用时可以在外部嵌入一个 checkbox-group 做统一的 checkbox 组，实现对于某一属性的多值选择。checkbox-group 的属性如表 4-8 所示。

checkbox 的属性如表 4-9 所示。

表 4-8 checkbox-group的属性

属 性 名	类 型	说 明
Bindchange	EventHandle	在<checkbox-group/>中选中项发生改变是触发 change 事件，detail = {value:[选中的 checkbox 的 value 的数组]}

表 4-9 checkbox的属性

属 性 名	类 型	默认值	说 明
Value	String		<checkbox/>标识，选中时触发<checkbox-group/>的 change 事件，并携带 <checkbox/>的 value
Disable	Boolean	false	是否禁用
Checked	Boolean	false	当前是否选中，可用来设置默认选中

【示例 4-7】

checkbox，新建项目命名为 demo_CheckBox，如图 4-12 所示。

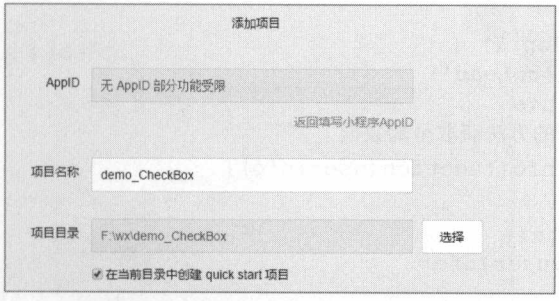


图 4-12 新建项目

编写页面文件 index.wxml：

```
<!--index.wxml-->
<view class="container">
  <checkbox-group  bindchange="checkboxChange">
   <label class="checkbox" wx:for="{{items}}">
    <checkbox value="{{item.name}}"/>{{item.value}}
  </label>
  </checkbox-group>
  <checkbox disabled="{{disabled}}">disabled</checkbox>
  <checkbox checked="{{checked}}">checked</checkbox>
</view>
```

编写 JS 中的 data 字段：

```
//index.js
```

```
//获取应用实例
var app = getApp()
Page({
  data: {
     items: [
      {name: '1', value: '选择 1'},
      {name: '2', value: '选择 2'},
      {name: '3', value: '选择 3'},
    ],
    disabled:true,
    checked:true,
  },
  //事件处理函数
  bindViewTap: function() {
    wx.navigateTo({
      url: '../logs/logs'
    })
  },
  onLoad: function () {
    console.log('onLoad')
    var that = this
    //调用应用实例的方法获取全局数据
    app.getUserInfo(function(userInfo){
      //更新数据
      that.setData({
        userInfo:userInfo
      })
    })
  }
})
```

显示效果如图 4-13 所示。

- 在主页面设置一个 checkbox-group，其中调用 for 方法循环一个 items 的变量。
- 在此 checkbox-group 中给每个 checkbox 赋值 items 的 name 和 value，动态生成 checkbox 节点。
- 第二行的 checkbox 是对于 check 的 disabled 属性的使用，使这个节点变得不可选择。
- 第三行的 checkbox 是对于 checked 属性的使用，默认勾选。



图 4-13　checkbox 显示效果

4.3.3 表单 form

小程序中的表单组件的作用与网页中表单的作用是一致的，都是将有关用户输入的内容，包括<switch/>、<input/>、<checkbox/>、<slider/>、<radio/>、<picker/>等元素提交到服务器。

与 Web 中表单一致的是其提交的键值对也是 name=>value 形式。开发者使用 name 作为其属性名，而用户所键入或者是默认的 value 将成为发送至服务器的一部分。

表单属性如表 4-10 所示。

表 4-10 表单属性

属 性 名	类 型	说 明
report-submit	Boolean	是否返回 formId 用于发送模板消息
bindsubmit	EventHandle	携带 form 中的数据触发 submit 事件，event.detail = {value : {'name': 'value'} , formId: ''}
bindreset	EventHandle	表单重置时会触发 reset 事件

【示例 4-8】

（1）新建项目，如图 4-14 所示。

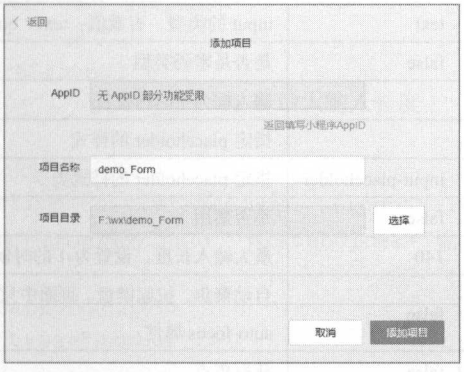


图 4-14 新建项目

（2）在视图中编辑代码：

```
<!--index.wxml-->
<view class="container">
  <form bindreset="formSubmit" bindreset="formRest">
    <input name="input" placeholder="在此输入" />
    <button type="primary" form-type="submit">提交</button>
    <button formType="reset">重置</button>
  </form>
</view>
```

示例效果如图 4-15 所示：

- 使用 form 表单组件包裹 input 元素，input 元素的 name 为 input，在图中键入数据“123”，那么单击“提交”按钮所提交的数值就是 input=>“123”。
- 表单中存在两个按钮，一个是绿色的“提交”按钮，一个是“重置”按钮，两个按钮分别监管两个不同的事件，一个为提交表单事件，一个为重置表单数据事件。



图 4-15　示例效果

4.3.4　输入框组件（input）

输入框组件用于表单的输入，在 form 组件使用 input，会出现一个可以输入的文本框。对于不同的应用需求，输入组件提供了不同的显示效果和输入方式。输入框组件的属性如表 4-11 所示。

表 4-11　输入框组件的属性

属 性 名	类　型	默 认 值	说　明
Value	String		输入框的初始内容
type	String	text	input 的类型，有效值：text、number、idcard、digit
password	Boolean	false	是否是密码类型
placeholder	String		输入框为空时占位符
placeholder-style	String		指定 placeholder 的样式
placeholder-class	String	input-placeholder	指定 placeholder 的样式类
disabled	Boolean	false	是否禁用
maxlength	Number	140	最大输入长度，设置为-1 的时候不限制最大长度
auto-focus	Boolean	false	自动聚焦，拉起键盘。页面中只能有一个<input/>或<textarea/>设置 auto-focus 属性
focus	Boolean	false	获取焦点
bindinput	EventHandle		除了 date/time 类型外的输入框，当键盘输入时，触发 input 事件，event.detail={value:value}，处理函数可以直接 return 一个字符串，将替换输入框的内容
bindfocus	EventHandle		输入框聚焦时触发，event.detail={value:value}
bindblur	EventHandle		输入框失去焦点时触发，event.detail={value:value}

【示例 4-9】

（1）新建项目，如图 4-16 所示。

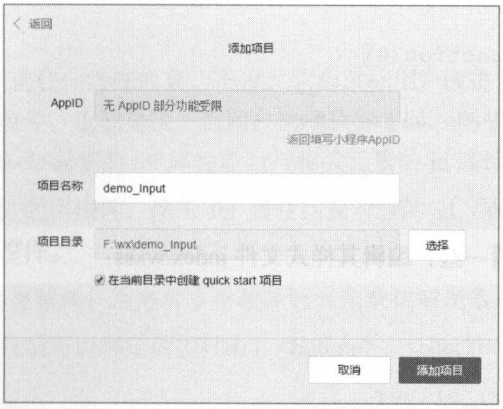


图 4-16 新建项目

（2）在 view 视图中编辑代码：

```
<!--index.wxml-->
<view class='myView'>
  <view class='myBackGround'>
    <input placeholder="这个是默认显示"/>
  </view>
  <view class='myBackGround2'>
    <input  maxlength="1" placeholder="最大输入长度 1" />
  </view>
  <view class='myBackGround'>
    <view class="section__title">输入的是：{{inputValue}}</view>
    <input  bindinput="bindKeyInput" placeholder="输入同步到 view 中" />
  </view>
    <view class='myBackGround2'>
    <input password type="number"/>
  </view>
  <view class='myBackGround'>
    <input placeholder-style="color:red" placeholder="占位符字体是红色的" />
  </view>
</view>
```

（3）在 JS 中编辑代码：

```
var app = getApp()
Page({
  data: {
    inputValue: ''
  },
```

```
  bindKeyInput: function(e) {
    this.setData({
      inputValue: e.detail.value
    })
  },
})
```

（4）为了让其样式好看一些，编辑其样式文件 index.wxss：

```
.myBackGround{
  height: 100%;
  width: 100%;
  background-color: silver
}
.myBackGround2{
  height: 100%;
  width: 100%;
  background-color: beige
}
.myView{
  margin:20%;
}
```

显示效果如图 4-17 所示。

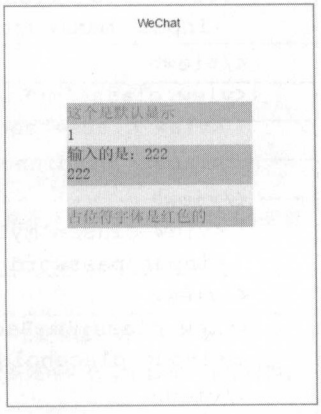


图 4-17　输入框显示效果

- 对于 index.wxml 中的第一个 input，使用了其属性 placeholder，这个属性是作为在未取得焦点的文本输入框中显示的默认提示值。
- 第二个 input 使用了 maxlength 属性，并且将其值设置成了 1，那么在其文本输入框中只能输入一位的字长，但是这个并不是文本框的 width 属性。
- 第三个 input 绑定了一个 JS 时间，也就是一个 JS 的 function，这个实现的效果是在输入文字时同步显示在其上面的 view 中。也就是在文本框输入时监听的事件，同步更改 view 中的值。
- 第四个 input 中使用了 placeholder-style 属性，这个属性是针对默认显示的提示文字的样式改写，同样，也可以使用 placeholder-class 属性对其样式进行整合后写在.wxss 文件中。

注意：部分样式在开发者工具中是不能出现效果的，所以如果有需要，需要在真实环境中进行测试。

4.3.5 label 元素

Label 元素不会向用户呈现任何特殊效果。不过，它为鼠标用户改进了可用性。如果在 label 元素内单击文本，就会触发此控件，也就是说，当用户选择该标签时，浏览器就会自动将焦点转到和标签相关的表单控件上。<label>标签的 for 属性应当与相关元素的 id 属性相同。

微信中用来改进表单组件的可用性，使用 for 属性找到对应的 id，或者将控件放在该标签下，当单击时，就会触发对应的控件。

注意：for 优先级高于内部控件，内部有多个控件的时候默认触发第一个控件。

在微信小程序开发中，目前可以绑定的控件有：<button/>、<checkbox/>、<switch/>。

4.3.6 滚动选择器 picker

picker 在微信小程序中是一种滚动选择器，到目前为止，它支持三种选择器，通过 mode 来区分不同的选择器。对于每一个选择器而言，其样式和内容用处已经是微信既定的了，主要实现一种向上或者向下的滚动选择效果。

picker 的三种选择器分别是普通选择器、时间选择器和日期选择器。

- 普通选择器（mode = selector）的属性如表 4-12 所示。

表 4-12 普通选择器的属性

属性名	类 型	默认值	说 明
Range	Array	[]	mode 为 selector 时，range 有效
Value	Number	0	mode 为 selector 时，是数字，表示选择了 range 中的第几个，从 0 开始
Bindchange	EventHandle		value 改变时触发 change 事件，event.detail={value:value}

- 时间选择器（mode=time）的属性和说明如表 4-13 所示。

表 4-13 时间选择器的属性

属 性 名	类 型	说 明
Value	String	表示选中的时间，格式为"hh:mm"
Start	String	表示有效时间范围的开始，字符串格式为"hh:mm"
End	String	表示有效时间范围的结束，字符串格式为"hh:mm"
bindchange	EventHandle	value 改变时触发 change 事件，event.detail = {value: value}

- 日期选择器（mode=date）的属性和说明如表 4-14 所示。

表 4-14　日期选择器的属性

属 性 名	类　型	默认值	说　明
Value	String	0	表示选中的日期，格式为"YYYY-MM-DD"
Start	String		表示有效日期范围的开始，字符串格式为"YYYY-MM-DD"
End	String		表示有效日期范围的结束，字符串格式为"YYYY-MM-DD"
fields	String	Day	有效值 year、month、day，表示选择器的粒度
bindchange	EventHandle		value 改变时触发 change 事件，event.detail = {value: value}

注意：开发工具暂时只支持 mode = selector。

【示例 4-10】

（1）新建项目，如图 4-18 所示。

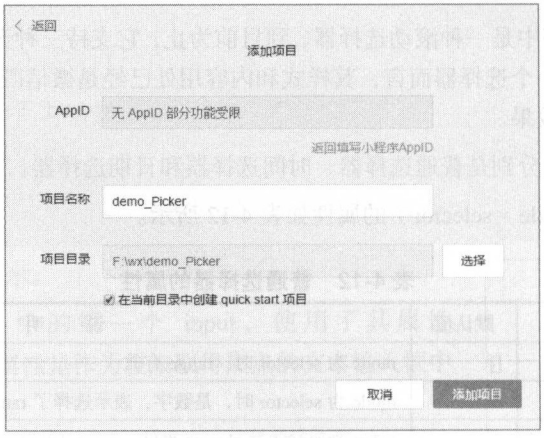


图 4-18　新建项目

（2）在 view 视图中编辑代码：

```
<view class="myView">
  <view class="title">选择器</view>
  <picker bindchange="bindPickerChange" value="{{index}}" range="{{array}}">
    <view class="picker">
      当前选择：{{array[index]}}
    </view>
  </picker>
</view>
<view class="myView">
  <view class="title">时间选择器</view>
  <picker   mode="time"   value="{{time}}"   start="09:01"   end="21:01"
```

```
bindchange="bindTimeChange">
      <view class="picker">
        当前选择: {{time}}
      </view>
    </picker>
  </view>

  <view class="myView">
    <view class="title">日期选择器</view>
    <picker mode="date" value="{{date}}" start="2016-11-30" end="2017-11-30"
bindchange="bindDateChange">
      <view class="picker">
        当前选择: {{date}}
      </view>
    </picker>
  </view>
```

（3）在 JS 中编辑代码：

```
  Page({
    data: {
      array: ['选择 1', '选择 2', '选择 3', '选择 4'],
      index: 0,
      date: '2016-11-30',
      time: '22:00'
    },
    bindPickerChange: function(e) {
      console.log('picker 发送选择改变，携带值为', e.detail.value)
      this.setData({
        index: e.detail.value
      })
    },
    bindDateChange: function(e) {
      this.setData({
        date: e.detail.value
      })
    },
    bindTimeChange: function(e) {
      this.setData({
        time: e.detail.value
      })
    }
  })
```

（4）更改相应的显示效果：

```
.myView{
  padding: 10px;
  padding-top: 10%;
}

.title{
  background-color: lemonchiffon
}
```

最终显示效果如图 4-19 所示。

- 第一个普通选择器，类似于一个下拉式的菜单选项，通过后台的 JS 控制其选中的内容显示在前台上。
- 第二个时间选择器，微信提供了一个 TimePicker 功能，用于获得时间，可以通过 start 和 end 控制其时间的初始时间和最晚时间。
- 第三个日期选择器，除了 start 和 end 的日期区间控制以外，还可以使用 fields 属性控制其时间选择的跨度。

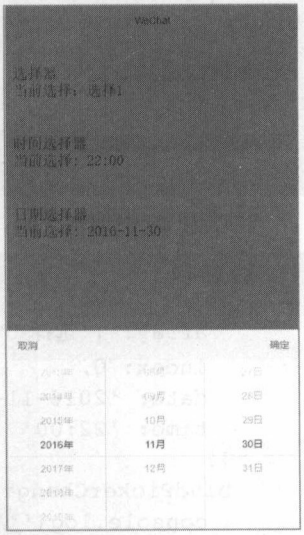


图 4-19　选择器显示效果

4.3.7　单选控制器（radio）

radio 是一个单项选择器，对于一整个 group（radio 组）而言，只可以选择其中一个选择项，所以称之为单项选择器。radio-group 作为一个单项选择器组，内部由多个<radio/>标签组成。radio-group 的属性如表 4-15 所示。

表 4-15　radio-group的属性

属性名	类　型	说　明
bindchange	EventHandle	<radio-group/>中的选中项发生变化时触发 change 事件，event.detail = {value: 选中项 radio 的 value}

单个的<radio/>标签也具有 radio 本身的属性，如表 4-16 所示。

表 4-16　radio的属性

属 性 名	类　型	默 认 值	说　明
Value	String		<radio/>标识。当该<radio/>选中时，<radio-group/>的 change 事件会携带<radio/>的 value
Checked	Boolean	false	当前是否选中
Disable	Boolean	false	是否禁用

【示例 4-11】

（1）新建项目，如图 4-20 所示。

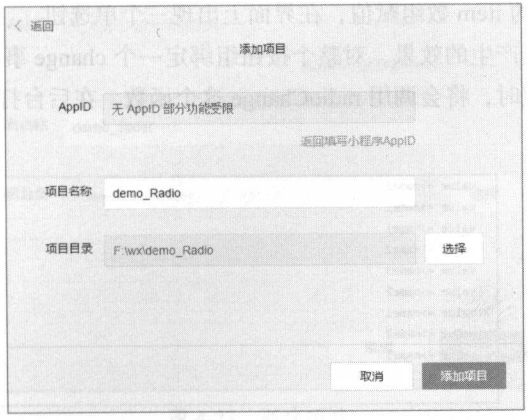


图 4-20 新建项目

（2）在 view 视图中编辑代码：

```
<view class="myView">
<radio-group class="radio-group" bindchange="radioChange">
  <label class="radio" wx:for="{{items}}">
    <radio       value="{{item.name}}"       checked="{{item.checked}}"/>
{{item.value}}
  </label>
</radio-group>
</view>
```

（3）在 JS 中编辑代码：

```
Page({
  data: {
    items: [
      {name: 'name1', value: 'name1'},
      {name: 'name2', value: 'name2', checked: 'true'},
      {name: 'name3', value: 'name3'},
    ]
  },
  radioChange: function(e) {
    console.log('value',"=>"+e.detail.value)
  }
})
```

本例显示效果如图 4-21 所示。

- 使用一个单选按钮组实现了三个单选 radio 的包裹。
- 通过后台的 JS 中的 item 数组赋值，在界面上出现三个单选钮。
- 为了清楚地看出其产生的效果，对整个按钮组绑定一个 change 事件进行 JS 控制台的输出，当改变选择的按钮时，将会调用 radioChange 这个函数，在后台打印输出现在的选择项，如图 4-22 所示。



图 4-21　单选控制器显示效果

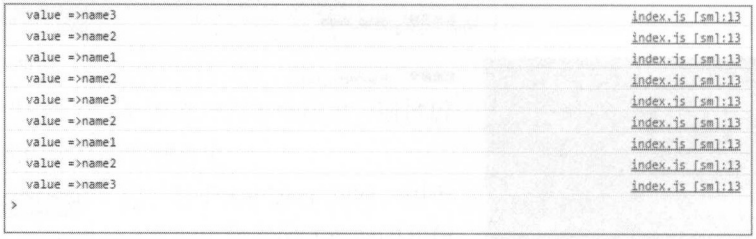


图 4-22　执行结果

4.3.8　滑动选择器（slider）

slider 显示一个可以滑动的控件，使得一些数值的输入形式变得更加人性化和细节化。滑动选择器可用于控制音乐的音量或控制时间长度。滑动选择器拥有表 4-17 所示的属性，对属性的不同赋值可以显示不同的效果。

表 4-17　slider的属性

属 性 名	类　型	默 认 值	说　明
min	Number	0	最小值
max	Number	100	最大值
step	Number	1	步长，取值必须大于 0，并且可被(max - min)整除
disabled	Boolean	false	是否禁用
value	Number	0	当前取值
show-value	Boolean	false	是否显示当前 value
bindchange	EventHandle		完成一次拖动后触发的事件，event.detail = {value: value}

【示例 4-12】

（1）新建项目，如图 4-23 所示。

（2）在 view 视图中编辑代码：

```
<view class="myView">
  <text>禁用</text>
```

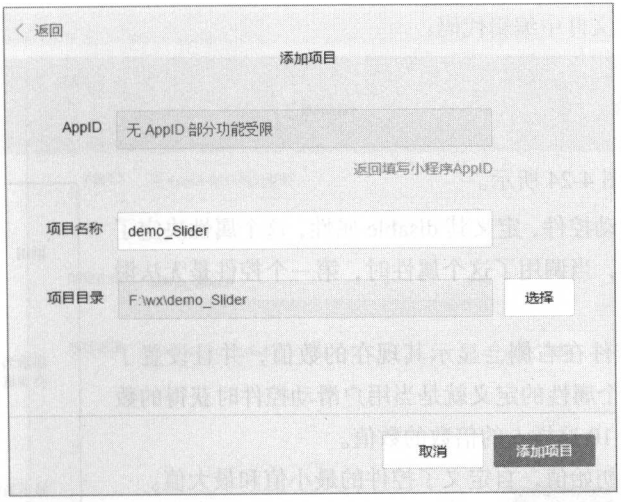


图 4-23 新建项目

```
  <view>
    <slider disabled/>
  </view>
</view>
<view class="myView">
  <text>显示当前 value，设置 step，滑动时会步进为 5</text>
  <view>
    <slider step="5" show-value/>
  </view>
</view>

<view class="myView">
  <text>显示当前 value 并且初始化值 80</text>
  <view>
    <slider value="80" show-value/>
  </view>
</view>

<view class="myView">
  <text>设置最小/最大值</text>
  <view>
    <slider min="50" max="200" show-value/>
  </view>
</view>
```

（3）在 wxss 样式文件中编辑代码：

```
.myView{
  padding: 10%
}
```

最终显示效果如图 4-24 所示。

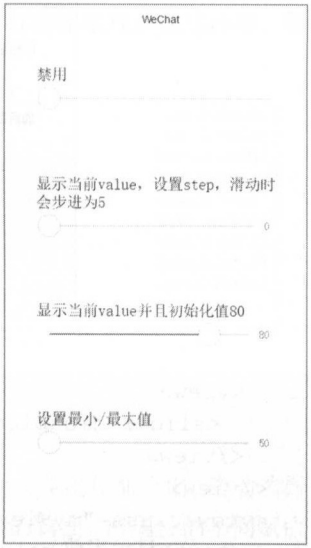


图 4-24　滑动选择器显示示例

- 对于第一个滑动控件，定义其 disable 属性，这个属性决定了其是否被禁用，当调用了这个属性时，第一个控件是无法滑动的。
- 第二个滑动控件在右侧会显示其现在的数值，并且设置了 step 属性，这个属性的定义就是当用户滑动控件时获得的数值就是 0、5、10 这样 5 的倍数的数值。
- 设定了控件的初始值，自定义了控件的最小值和最大值。

注意：对于很多控件中的属性，默认值都为 false 时，只需要在控件中使用其属性，便相当于将其值更改为 true。

4.3.9　开关选择器（switch）

Switch 控件是微信小程序中用来提供用户开关效果的一组控件，通过不同的显示效果，达到视觉上的开启和关闭的效果，这种滑动式的开关最早用于苹果手机。

switch 实现的效果等同于单一的 checkbox 控件，但外观上和表达上有着不一样的效果，其属性如表 4-18 所示。

表 4-18　switch属性

属 性 名	类　型	默 认 值	说　明
Checked	Boolean	false	是否选中
Type	String	switch	样式，有效值：switch、checkbox
Bindchange	EventHandle		checked 改变时触发 change 事件，event.detail={ value:checked}

【示例 4-13】

（1）新建项目，如图 4-25 所示。

（2）在 view 视图中编辑代码：

```
<view class="myView">
    <switch checked bindchange="switch1Change"/>
    <switch bindchange="switch2Change"/>
```

```
</view>
```



图 4-25 新建项目

（3）在 JS 中编辑代码：

```
Page({
  switch1Change: function (e){
    console.log('单击 switch1', e.detail.value)
  },
  switch2Change: function (e){
    console.log('单击 switch2', e.detail.value)
  }
})
```

（4）更改 wxss 样式文件，让整个视图变得更美观：

```
.myView{
  padding: 35%;
  padding-top: 70%
}
```

本例最终显示效果如图 4-26 所示。

- 第一个控件默认选中是 true，所以如图 4-26 可以看到其效果是选中状态，同时，其值也是 true。
- 对于两个控件，分别在其改变的时候设置了两个监听事件，当对其状态进行改变时，会在控制台上打印输出其改变后的值。单击后执行效果如图 4-27 所示。

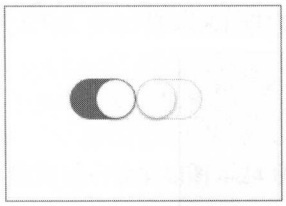


图 4-26　开关选择器显示效果

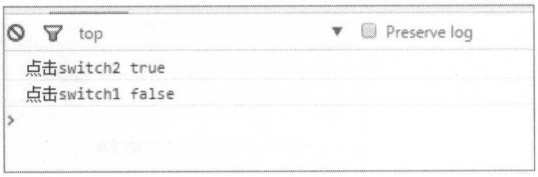


图 4-27　执行效果

4.3.10　文本框控件（textarea）

文本框控件是微信小程序中用于长文本输入的控件，通常在 form 表单中输入备注或者文章等大量的文字时使用。textarea 的属性如表 4-19 所示。

表 4-19　textarea的属性

属 性 名	类　型	默 认 值	说　明
value	String		输入框的内容
placeholder	String		输入框为空时占位符
placeholder-style	String		指定 placeholder 的样式
placeholder-class	String	textarea-placeholder	指定 placeholder 的样式类
disabled	Boolean	false	是否禁用
maxlength	Number	140	最大输入长度，设置为 0 的时候不限制最大长度
auto-focus	Boolean	false	false 自动聚焦，拉起键盘。页面中只能有一个<textarea/>或<input/>设置 auto-focus 属性
focus	Boolean	false	获取焦点（开发工具暂不支持）
auto-height	Boolean	false	是否自动增高，设置 auto-height 时，style.height 不生效
bindfocus	EventHandle		输入框聚焦时触发，event.detail={value:value}
bindblur	EventHandle		输入框失去焦点时触发，event.detail={value:value}
bindlinechange	EventHandle		输入框行数变化时调用 event.detail={height:0, heightRpx:0,lineCount:0}

【示例 4-14】

（1）新建项目，如图 4-28 所示。

（2）在 view 视图中编辑代码：

```
  <view class="myView">
    <textarea     bindfocus="bindGetFocus"          bindblur="bindLostFocus"
indlinechange="bindLineChanged" auto-height placeholder="文本框" />
  </view>
```

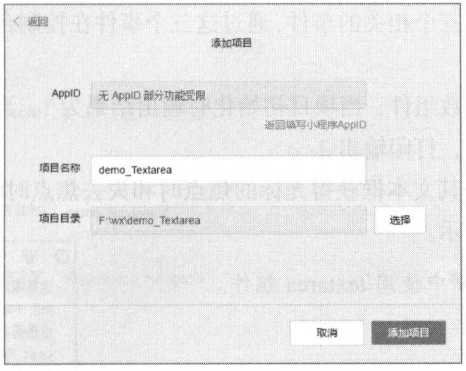


图 4-28 新建项目

（3）在 JS 中编辑代码：

```
var i=0
Page({
  bindGetFocus: function() {
    console.log("Get Focus")
  },
  bindLostFocus: function(e) {
    console.log("Lost Focus")
  },
  bindLineChanged:function(e){
    i=i+1
    console.log("这是第"+i+"行")
  }
})
```

（4）更改 wxss 样式文件，让整个视图变得更美观：

```
.myView{
  padding: 10%;
  padding-top: 40%
}
```

本例显示效果如图 4-29 所示。

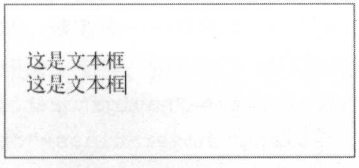


图 4-29 文本显示效果

- 示例代码主要绑定了三个相关的事件，通过这三个事件在控制台的响应输出来辅助理解其控件属性。
- 第一个事件是响应行数组件，当项目初始化时输出结果为 1，当输入一个换行符或者字长超过第一行的长度之后，打印输出 2。
- 其他两个事件分别是其文本框获得光标的焦点时和失去焦点时响应的事件，分别在控制台打印输出一句相应的提示。

注意：请勿在 scroll-view 中使用 textarea 组件。

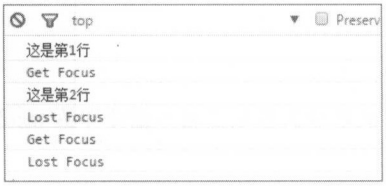


图 4-30　执行结果

4.4　导航

本节介绍的导航组件是小程序中相当于导航标签一类的组件，它可以让用户的操作从单一的界面变成丰富的不同页面，使整个小程序代码更加合理且结构清晰。对于小程序的结构而言，只拥有一个页面是不合理的，而导航组件搭建了页面与页面之间的桥梁。

页面链接（navigator）控件相当于页面中的超链接<a>标签，其只支持小程序中的内部页面跳转。页面链接控件存在两种不同的打开新页面的形式：

- 一种是相当于实例化一个完全崭新的页面。
- 另一种是在打开新页面的同时关闭旧页面，即相当于在当前页面打开这个页面链接。

页面链接的属性如表 4-20 所示。

表 4-20　页面链接的属性

属 性 名	类　型	默 认 值	说　明
url	String		应用内的跳转链接
redirect	Boolean	false	是否关闭当前页面
hover-class	String	navigator-hover	指定单击时的样式类，当 hover-class="none"时，没有单击效果

【示例 4-15】

（1）新建项目，如图 4-31 所示。

（2）在 view 视图中编辑代码：

```
<view class="btn-area">
  <navigator url="test" hover-class="navigator-hover">跳转到新页面</navigator>
  <navigator url="test" redirect hover-class="other-navigator- hover">在当前页打开</navigator>
</view>
```



图 4-31 新建项目

（3）在 JS 中编辑代码：

```
Page({
  onLoad: function(options) {
    this.setData({
      title: options.title
    })
  }
})
```

链接显示效果如图 4-32 所示。



图 4-32 链接显示效果

（4）如果要实现页面间的跳转，就需要一个新的页面，此时需要在 app.json 这个文件中进行新页面的注册，在编辑窗口双击打开这个文件，添加新页面注册代码。

```
{
  "pages":[
    "pages/index/index",
    "pages/index/test",
```

```
    "pages/logs/logs"
  ],
  "window":{
    "backgroundTextStyle":"light",
    "navigationBarBackgroundColor": "#fff",
    "navigationBarTitleText": "WeChat",
    "navigationBarTextStyle":"black"
  }
}
```

输入新页面之后，微信开发者工具会自动建立此页面的三个页面文件，并且会自动保存在需要的文件夹中。

（5）在 test.wxml 中键入以下代码用于测试：

```
<view>
    <text>测试</text>
</view>
```

保存后单击 index 页面中的两个链接会发现单击第一个新页面链接时会在左上角出现一个返回按钮，而单击第二个链接，在当前页面打开链接的选项时，不会有这个按钮，如图 4-33 和图 4-34 所示。



图 4-33　保留前页显示方式



图 4-34　直接跳转方式

注意：navigator-hover 默认为{background-color: rgba(0, 0, 0, 0.1); opacity: 0.7;}。

4.5　媒体组件

本节介绍的媒体组件，顾名思义，就是微信小程序中的多媒体部分，通过相关的媒体组件可以在小程序中体验到包括但不限于音乐、图片、视频等多媒体的传播形式。通过对这些组件的组合和应用，可以生成一个在线播放音乐乃至于视屏播放器的微信小程序。

4.5.1 音频组件（audio）

音频组件可以实现音频类的在线播放和操作控制。甚至于微信官方已经构想到了不错的样式和操作方式，可以非常简单地实现一个不错的用户 UI。audio 的属性如表 4-21 所示。

表 4-21 audio的属性

属 性 名	类 型	默 认 值	说 明
id	String		video 组件的唯一标识符
src	String		要播放音频的资源地址
loop	Boolean	false	是否循环播放
controls	Boolean	true	是否显示默认控件
poster	String		默认控件上的音频封面的图片地址，controls 属性值为 false，则设置 poster 无效
name	String	未知音频	默认控件上的音频名字，如果 controls 属性值为 false，则设置 name 无效
author	String	未知作者	默认控件上的作者名字，如果 controls 属性值为 false，则设置 author 无效
binderror	EventHandle		当发生错误时触发 error 事件，detail={errMsg:MediaError.code}
bindplay	EventHandle		当开始/继续播放时触发 play 事件
bindpause	EventHandle		当暂停播放时触发 pause 事件
bindtimeupdate	EventHandle		当播放进度改变时触发 timeupdate 事件，detail={currentTime, duration}
bindended	EventHandle		当播放到末尾时触发 ended 事件

如果 binderror 属性出现错误时，将会显示不同的错误信息，以供开发者获取错误信息，并提供相应的解决方案给用户。MediaError.code 一共有四种不同的错误提示，具体的错误代码和说明如表 4-22 所示。

表 4-22 错误代码和说明

返回错误码	描 述
MEDIA_ERR_ABORTED	获取资源被用户禁止
MEDIA_ERR_NETWORD	网络错误
MEDIA_ERR_DECODE	解码错误
MEDIA_ERR_SRC_NOT_SUPPOERTED	不合适资源

注意：开发时一般不会出现这些错误提示，但这种错误也是无法避免和阻止的，最佳的解决办法就是给用户一个友好的提示，让用户明白无法播放的原因是权限或者网络问题，而非程序本身。

4.5.2 上手实践：音乐播放器

（1）新建项目，如图 4-35 所示。



图 4-35　新建项目

（2）在 view 视图中编辑代码：

```
<view class="myView">
  <audio  poster="{{poster}}"  name="{{name}}"  author="{{author}}"
src="{{src}}" id="myAudio" controls loop></audio>
</view>
<view class="myBtn">
  <button class="btn" type="primary" bindtap="audioPlay">播放</button>
  <button class="btn" type="warn" bindtap="audioPause">暂停</button>
  <button class="btn" type="default" bindtap="audio30">跳至 30 秒</button>
</view>
```

（3）在 JS 中编辑代码，这里使用了网易云音乐的 API 拿到 mp3 地址，并且为了使界面更美观，增加专辑封面，图片来自百度。

```
Page({
  onReady: function (e) {
    //使用 wx.createAudioContext 获取 audio 上下文 context
    this.audioCtx = wx.createAudioContext('myAudio')
  },
  data: {
    poster:          'http://hiphotos.baidu.com/%B9%AB%B6%FA/pic/item/
6d617b1b8701a18b4e04b9fb9e2f07082a38fed1.jpg',
    name: '身骑白马',
    author: '徐佳莹',
    src:          'http://m2.music.126.net/9Py8jhtwE3vDfV_regBavQ==/
5964850581190727.mp3',
```

```
  },
  audioPlay: function () {
    this.audioCtx.play()
  },
  audioPause: function () {
    this.audioCtx.pause()
  },
  Audio30: function () {
    this.audioCtx.seek(30)
  }
})
```

（4）更改 wxss 样式文件，让整个视图变得更美观：

```
.myView{
  padding: 10%;
  padding-top: 40%
}

.myBtn{
  padding: 10px
}
.btn{
  width: 30%;
  float: left;
  margin-left: 10px
}
```

最终显示效果如图 4-36 所示。



图 4-36 媒体标签显示效果

- 通过设置<audio>的音频地址（src）和专辑图片的地址（poser），载入网络上的音频和封面信息，形成了一个矩形框的音乐播放样式，单击专辑封面的“播放”按钮可以进行音乐的播放。
- 通过对于 author 和 name 的设置确定了在右方显示的音乐名称和作者，而对于整个 audio 控件，最右边会自动加载其播放的进度。
- 下方三个按钮的作用就是对于音频播放的控制和调整，先通过 this.audioCtx = wx.createAudioContext('myAudio')获取播放控件的上下文（这里其实相当于网页的 getElementbyId），再通过监听单击按钮事件来控制属性。

注意：audio 控件不支持部分音频格式，请在开发前确定音源的可播放性。

4.5.3 图片控件（image）

图片控件在微信小程序中展示不同的图片，还可以对图片进行编辑，通过不同的属性可以对图片进行切割或缩放。image 的属性如表 4-23 所示。

表 4-23 image的属性

属性名	类型	默认值	说明
src	String		图片资源地址
mode	String	scaleToFill	图片裁剪、缩放的模式
binderror	HandleEvent		当错误发生时，发布到 AppService 的事件名，事件对象 event.detail={errMsg:'somethingwrong'}
bindload	HandleEvent		当图片载入完毕时，发布到 AppService 的事件名，事件对象 event.detail={height:'图片高度 px',width:'图片宽度 px'}

表 4-23 中的属性 mode 拥有 12 种对应的模式，其中 3 种是缩放模式，9 种是裁剪模式。具体模式如表 4-24 所示

表 4-24 mode的具体模式

模式	说明
scaleToFill	不保持纵横比缩放图片，使图片的宽高完全拉伸至填满 image 元素
aspectFit	保持纵横比缩放图片，使图片的长边能完全显示出来。也就是说，可以完整地将图片显示出来
aspectFill	保持纵横比缩放图片，只保证图片的短边能完全显示出来。也就是说，图片通常只在水平或垂直方向是完整的，另一个方向将会发生截取
top	不缩放图片，只显示图片的顶部区域
bottom	不缩放图片，只显示图片的底部区域
center	不缩放图片，只显示图片的中间区域
left	不缩放图片，只显示图片的左边区域
right	不缩放图片，只显示图片的右边区域
top left	不缩放图片，只显示图片的左上边区域
top right	不缩放图片，只显示图片的右上边区域
bottom left	不缩放图片，只显示图片的左下边区域
bottom right	不缩放图片，只显示图片的右下边区域

4.5.4 上手实践：相册

（1）新建项目，如图 4-37 所示。



图 4-37 新建项目

（2）在 view 视图中编辑代码，这里使用一个简单的循环，在 JS 文件中控制相应的显示效果：

```
<view class="myView">
  <view class="myDemo">
    <view wx:for="{{array}}" wx:for-item="item">
      <view>{{item.text}}</view>
      <view class="myPic">
        <image style="width: 200px; height: 200px; background-color: black;"
mode="{{item.mode}}" src="{{src}}"></image>
      </view>
    </view>
  </view>
</view>
```

（3）在 JS 中编辑代码，这里使用了百度中的一张动态的 png 图片作为统一展示地址：

```
Page({
  data: {
    array: [{
      mode: 'scaleToFill',
      text: 'scaleToFill: 不保持纵横比缩放图片，使图片完全适应'
    }, {
      mode: 'aspectFit',
      text: 'aspectFit: 保持纵横比缩放图片，使图片的长边能完全显示出来'
    }, {
      mode: 'aspectFill',
      text: 'aspectFill: 保持纵横比缩放图片，只保证图片的短边能完全显示出来'
```

```
    }, {
      mode: 'top',
      text: 'top: 不缩放图片，只显示图片的顶部区域'
    }, {
      mode: 'bottom',
      text: 'bottom: 不缩放图片，只显示图片的底部区域'
    }, {
      mode: 'center',
      text: 'center: 不缩放图片，只显示图片的中间区域'
    }, {
      mode: 'left',
      text: 'left: 不缩放图片，只显示图片的左边区域'
    }, {
      mode: 'right',
      text: 'right: 不缩放图片，只显示图片的右边边区域'
    }, {
      mode: 'top left',
      text: 'top left: 不缩放图片，只显示图片的左上边区域'
    }, {
      mode: 'top right',
      text: 'top right: 不缩放图片，只显示图片的右上边区域'
    }, {
      mode: 'bottom left',
      text: 'bottom left: 不缩放图片，只显示图片的左下边区域'
    }, {
      mode: 'bottom right',
      text: 'bottom right: 不缩放图片，只显示图片的右下边区域'
    }],
    src: 'http://img.mp.itc.cn/upload/20161102/4039a02341284a36b8dd271609cbf53e_th.gif'
  },
})
```

（4）更改 wxss 样式文件，让整个视图变得更美观：

```
.myView{
  padding: 20px
}
.myDemo{
  font-size: 15px
}
.myPic{
```

```
    padding: 20%
}
```

最终显示效果如图 4-38 和图 4-39 所示。

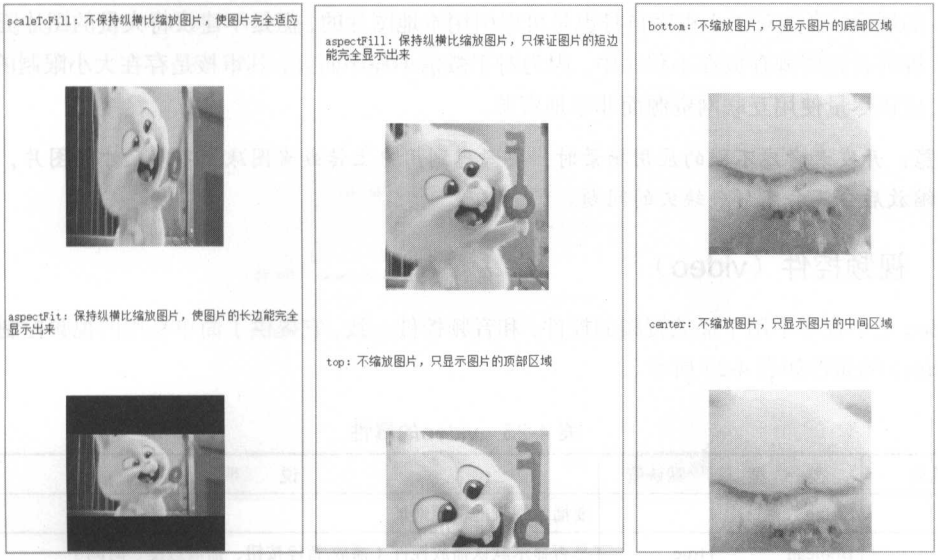


图 4-38　image 不同显示示例 1

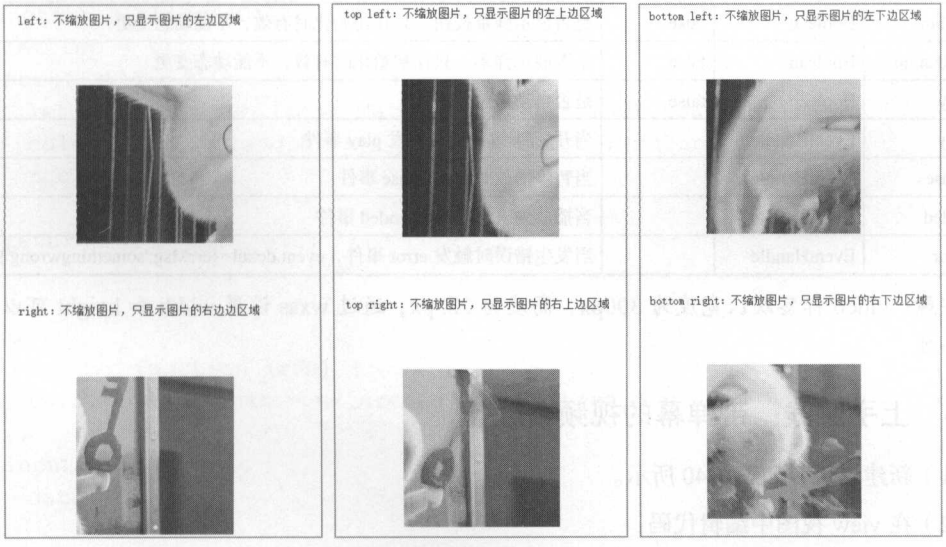


图 4-39　image 不同显示示例 2

- 在 JS 的控制文件中创立一个相关的数组，包含不同的 mode 属性、用于说明的 title、统一使用的 src。
- 开发者可以通过自己实现后台服务器（或者是相关的云服务）拿到相关的图片 url 进行图片的展示，当然对于小程序自身也是可以引用本地图片的，但是不建议将大量的图片资源或者视屏音频资源存放在小程序中，因为对于微信小程序而言，其审核是存在大小限制的，所以应该尽量使用互联网资源而非本地资源。

注意：开发者考虑不同的应用场景时，要考虑到用户上传或者图床上不同尺寸的图片，在经过统一的缩放后会不会变形或缺失的问题。

4.5.5 视频控件（video）

video 是小程序中用于播放视频的控件，和音频控件一致，它提供了简单易用的视频控制插件和 API。video 的属性如表 4-25 所示。

表 4-25 video的属性

属性名	类 型	默认值	说 明
src	String		要播放视频的资源地址
controls	Boolean	true	是否显示默认播放控件（播放/暂停按钮、播放进度、时间）
danmu-list	Object Array		弹幕列表
danmu-btn	Boolean	false	是否显示弹幕按钮，只在初始化时有效，不能动态变更
enable-danmu	Boolean	false	是否展示弹幕，只在初始化时有效，不能动态变更
autoplay	Boolean	false	是否自动播放
bindplay	EventHandle		当开始/继续播放时触发 play 事件
bindpause	EventHandle		当暂停播放时触发 pause 事件
bindended	EventHandle		当播放到末尾时触发 ended 事件
binderror	EventHandle		当发生错误时触发 error 事件，event.detail={errMsg:'somethingwrong'}

注意：video 标签默认宽度为 300px，高度为 225px，通过 wxss 设置 width 和 height 可以调整宽度和高度。

4.5.6 上手实践：带弹幕的视频播放器

（1）新建项目，如图 4-40 所示。

（2）在 view 视图中编辑代码：

```
<view class="myView">
  <video id="myVideo" src="" binderror="videoErrorCallback" danmu-list=
```

```
"{{danmuList}}" enable-danmu danmu-btn controls></video>
    <view class="myDanmu">
      <input class="DanmuInput" bindblur="bindInputBlur"/>
      <button class="SendDanmu" bindtap="bindSendDanmu">发送弹幕</button>
    </view>
  </view>
```



图 4-40 新建项目

（3）在 JS 中编辑代码，这里定义了一个 getRandomColor 方法，用来区分不同弹幕的颜色：

```
function getRandomColor () {
  let rgb = []
  for (let i = 0 ; i < 3; ++i){
    let color = Math.floor(Math.random() * 256).toString(16)
    color = color.length == 1 ? '0' + color : color
    rgb.push(color)
  }
  return '#' + rgb.join('')
}

Page({
  onReady: function (res) {
    this.videoContext = wx.createVideoContext('myVideo')
  },
  inputValue: '',
    data: {
       src: '',
    danmuList: [
      {
```

```
        text: '弹幕 1',
        color: '#ff0000',
        time: 1
      },
      {
        text: '弹幕 2',
        color: '#ff00ff',
        time: 3
      }
    ]
    },
  bindInputBlur: function(e) {
    this.inputValue = e.detail.value
  },
  bindSendDanmu: function () {
    this.videoContext.sendDanmu({
      text: this.inputValue,
      color: getRandomColor()
    })
  },
})
```

（4）更改 wxss 样式文件，让整个视图变得更美观：

```
.myView{
  padding: 30px;
  padding-top: 30%;
}
.myDanmu{
  padding: 10px;
}
.DanmuInput{
  float: left;
  width: 50%;
  background-color: lightgray
}
.SendDanmu{
  float: left;
  width: 40%;
  font-size: 12px;
  margin-left:5px
}
```

显示效果如图 4-41 所示，单击“播放”按钮可以播放在线视频，弹幕按钮可以关闭或者开启弹幕选项，在下方的弹幕输入框中输入想要发送的内容，就可以发送弹幕了。

- 使用 video 标签引入一个在线的视频和自带的播放器，就可实现视频的播放操作。
- 在 JS 文件中定义一个数组用来模拟存储已有的弹幕信息，同时在发送按钮上绑定监听事件，当用户单击按钮的时候调用 sendDanmu 发送即时的弹幕信息。

注意：对于联网的应用可以通过 API 拿到相关的弹幕信息，并且在 JS 中对弹幕进行格式化处理，交由 wxml 显示。而对用户实时输入的弹幕信息，除了在视频上显示以外，还需要通过相应的 API 上传到服务器中进行持久化的存储。

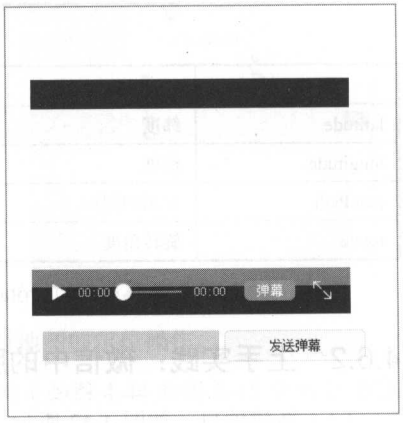


图 4-41 视频显示效果

4.6 地图

微信小程序中的地图（map）控件主要是为了解决小程序的地理位置定位和导航等应用场景。其本质是结合腾讯地图 API 去确定一个地理位置点，用作简单的显示效果和导航功能。

4.6.1 地图

地图控件已经结合了腾讯地图 API，所以并不需要引入外部 JS 和大量的方法，只需要通过属性控制和内容填充就可以实现地图控件的功能。map 的属性如表 4-26 所示。

表 4-26 map的属性

属 性 名	类 型	默 认 值	说 明
longitude	Number		中心经度
latitude	Number		中心纬度
scale	Number	16	缩放级别
markers	Array		标记点
covers	Array		覆盖物

标记点用于在地图上显示标记的位置，不能自定义图标和样式，但是可以对其定位和表述进行更改。标记点的属性如表 4-27 所示。

注意：这四个属性是必填的。

不同的地点存在另一个可以被定义的图标，也就是覆盖物。覆盖物的属性如表 4-28 所示。

表 4-27　标记点的属性

属　性	说　明	类　型	备　注
latitude	纬度	Number	浮点数，范围-90~90
longitude	经度	Number	浮点数，范围-180~180
name	标注点名	String	
desc	标注点详细描述	String	

表 4-28　覆盖物的属性

属　性	说　明	类　型	备　注
latitude	纬度	Number	浮点数，范围-90~90
longitude	经度	Number	浮点数，范围-180~180
iconPath	显示的图标	String	项目目录下的图片路径，支持相对路径写法
rotate	旋转角度	Number	顺时针旋转的角度，范围 0~360，默认为 0

注意：对于覆盖物属性除了 rotate，其他属性也是必填项。

4.6.2　上手实践：微信中的腾讯地图定位

（1）新建项目，如图 4-42 所示。

（2）在 view 视图中编辑代码：

```
<view class="container">
  <map longitude="113.324520" latitude="23.099994" markers=" {{markers}}"
covers="{{covers}}" style="width: 100%; height: 300px;"></map>
</view>
```

（3）在 JS 中编辑代码，这里定位到北京的天安门部分，用户通过拖动地图和选择地点可以使用导航功能：

```
Page({
  data: {
    markers: [{
      latitude: 39.9139570000,
      longitude: 116.4040560000,
      name: '我爱祖国天安门',
      desc: '天安门'
    }],
  }
})
```

本例显示效果如图 4-43 所示。

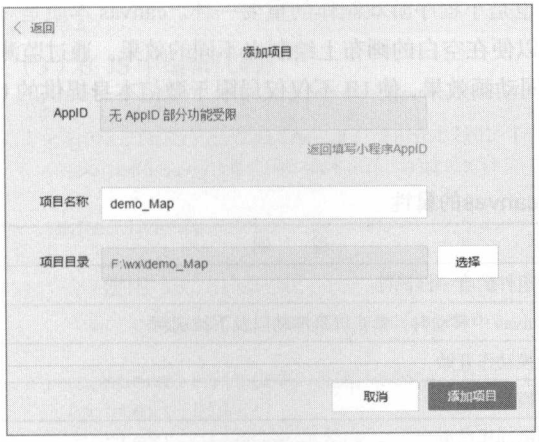


图 4-42 新建项目



图 4-43 地图显示效果

这里使用了地图控件，通过 markers 的属性定义，实现了对地图的定位操作。

注意：地图操作是可以单击“去这里”按钮进行导航操作的（如图 4-44 和图 4-45 所示），但在开发者工具里无法实现自动定位。

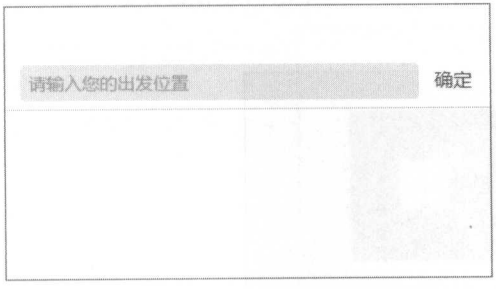


图 4-44 搜索地理位置



图 4-45 “去这里”

4.7 画布

微信小程序中的画布是非常重要的展示控件，其相当于 HTML 中的 canvas 控件。开发者可以在这个控件中绘制想要的图形，甚至于实现一些动画和数据展示效果。

4.7.1 画布控件（canvas）

画布控件用于数据展示或动态效果展示，它也是小程序游戏制作的重要一环。canvas 本质是一块空白的画布，可以通过程序代码进行画笔控制，以便在空白的画布上绘制出不同的效果。通过监测用户单击事件以及不同状态，可以在画布上绘制不同动画效果，使 UI 不仅仅局限于微信本身提供的 UI。

canvas 的属性如表 4-29 所示。

表 4-29 canvas的属性

属 性 名	类 型	默 认 值	说 明
canvas-id	String		canvas 组件的唯一标识符
disable-scroll	Boolean	false	当在 canvas 中移动时，禁止屏幕滚动以及下拉刷新
bindtouchstart	EventHandle		手指触摸动作开始
bindtouchmove	EventHandle		手指触摸后移动
bindtouchend	EventHandle		手指触摸动作结束
bindtouchcancel	EventHandle		手指触摸动作被打断，如来电提醒、弹窗
binderror	EventHandle		当发生错误时触发 error 事件，detail={errMsg:'somethingwrong'}

注意：canvas 标签默认宽度为 300px，高度为 225px，可通过 wxss 进行更改。对于同一页面而言，canvas-id 不可重复，如果使用一个已经出现过的 canvas-id，该 canvas 标签对应的画布将被隐藏并不再正常工作。

4.7.2 上手实践：使用 canvas 实现画布的绘制

（1）新建项目，如图 4-46 所示。



图 4-46 新建项目

（2）在 view 视图中编辑代码：

```
<canvas style="width: 300px; height: 200px;" canvas-id="firstCanvas">
</canvas>
```

（3）在 JS 中编辑代码：

```
Page({
  canvasIdErrorCallback: function (e) {
    console.error(e.detail.errMsg)
  },
  onReady: function (e) {

    //使用 wx.createContext 获取绘图上下文 context
    var context = wx.createContext()

    context.rect(50, 50, 200, 200)
    context.fill()
    context.clearRect(100, 100, 50, 50)
    //调用 wx.drawCanvas，通过 canvas id 指定在哪张画布上绘制，通过 actions 指定绘制行为
    wx.drawCanvas({
      canvasId: 'firstCanvas',
      actions: context.getActions() //获取绘图动作数组
    })
  }
})
```

本例显示效果如图 4-47 所示。

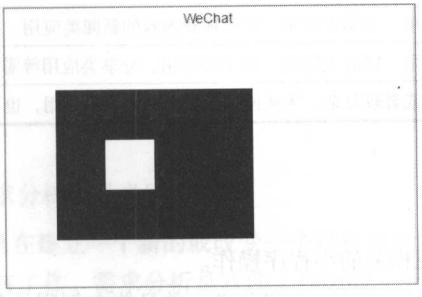


图 4-47 画布显示效果

- 在页面中引入画布控件，并对其大小进行限定。
- 在 JS 代码中通过获取到画笔，进行绘制图像，首先绘制一个矩形，坐标点为（50,50,200,200）并且对其进行填充。
- 之后对于（100,100,50,50）内的矩形进行清除，露出底色。

4.8 小结

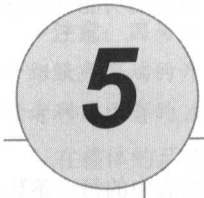
本章讲解了微信小程序的组件部分，通过这些组件，可以让小程序应用从本身单一的展现方式变得丰富起来。对每个控件的使用场景读者可以灵活地选择和组合。一些基础组件的应用场景如表4-30所示。

表 4-30 基础组件的应用场景

组件名称	使用场景
视图容器（view）	基础小程序组件，对于任何的小程序应用都是不可缺少的组件之一，也是使用最频繁的组件之一
可滚动视图区域（scroll-view）	文章发布类 App、新闻类 App、内容类 App 等
滑块视图容器（swiper）	需要滑动效果或者是选择项目较多无法在单屏显示相关效果，又无法分页的 App
图标（icon）	需要对于用户做出提示或者是输入验证反馈的应用场景，提供相关的标识，增强和用户的交流
文本（text）	基础小程序组件，对于任何的小程序应用都是不可缺少的组件之一，也是使用最频繁的组件之一
进度条（progress）	显示下载进度或者是载入进度的应用场景，可以用于资源载入时的过度场景
表单组件	包括 button、checkbox 等相关的输入框、文本框、按钮等组件，用于用户输入信息或者是展示相关信息的应用场合
页面链接（navigator）	常用组件之一，它是除去单页的简单应用都应该使用的组件之一，用于页面的跳转等场景
音频组件（audio）	音乐类应用、录音类应用、社交类应用、分享类应用必备组件，甚至是内容类应用也可以使用
图片组件（image）	文章发布类 App、新闻类 App、内容类 App 等，也是最常用的组件之一
视频组件（video）	视频类应用、分享类应用，需要视频内容的新闻类应用、展示类应用等需要视频播放的场合
地图组件（map）	导航类应用、LBS 类应用、社交类应用、分享类应用等需要位置信息和地图类信息的应用
画布（canvas）	需要绘图或者较复杂、无法使用原始组件实现的应用，也可用于入场切换动画效果

4.9 实践与练习

- 根据每一节的内容实现相关的小程序操作。
- 理解章节内部示例程序的代码结构和含义，并且能在例程的基础上进行衍生和重构。
- 理解并总结微信小程序和网页开发的不同点和相似处。
- 深入理解每一个控件的使用场景和使用方式，能通过不同的组合去重构市面上几乎所有的应用 App。



项目实战 1——日记本

第 2 章和第 3 章介绍了微信小程序开发中的一些程序基础，第 4 章介了绍微信小程序的基础 API。本章通过一个项目（联网交互的小程序）的制作流程，带领读者真正步入到小程序的开发中来。

本章学习目标：

- 通过 API 进行微信小程序的开发，并且能保证运行。
- 了解一个微信小程序的整体设计思路和模式。
- 在本章的基础上进行思考，并且能举一反三。

注意：本章会使用一些常用的软件开发方式进行讲解，同时可能会涉及第 6 章的部分内容，或未在本书范围内的一些内容，如果读者有需求，可以深挖技术深入学习。

5.1 需求分析

5.1.1 需求确定

在开发任何一个产品时需求分析都是必经阶段。

在软件工程中，需求分析是在建立一个新的或改变一个现存的电脑系统时，描写新系统的目的、范围、定义和功能所要做的所有工作。需求分析是软件工程中的一个关键过程，在此过程中，系统分析员和软件工程师确定顾客的需要，只有在确定了这些需要后，他们才能够分析和寻求新系统的解决方法。

需求分析阶段的任务是确定软件系统功能。在本章中，开发例子的需求是一个假设的需求——一个联网的个人日记系统。

5.1.2 功能确定

有了明确的需求，再对日记系统进行一系列的分析和功能设计。对一个确定需求的功能分析过程，是将确定的需求进行功能化，而非只是提供一个大致的概念。

那么，一个简单的个人日记系统需要怎样的一些功能呢？

作为日记系统，最重要也最基础的功能，是日记的书写和存储功能。

对已经存储的日记进行查看的功能，可以查看所有的日记列表，并且可以对列表中的每一条数据进行完整内容的查看，同时为了完善功能，对于已经写过并保存的日记应该能进行删除等操作。

既然是联网型的应用，那么应对每个不同的用户进行区分，简单来说，这里需要用户进行注册和登录。用户登录后可对个人的日记内容进行更改，但对他人的日记则不能进行操作。同时可以在日记中增加一个可以被共享或者只有个人可见的选项。

在不考虑将来的扩展性开发等功能的情况下，暂时将上述确定的功能进行一个汇总，作为此微信小程序需要完成的功能。

个人日记系统：

- 基础的日记书写、上传功能。
- 对于日记书写可以分类为公开和个人。
- 基础的用户注册功能。
- 基础的用户登录功能。
- 对于个人日记列表的读取功能。
- 对于个人日记的读取详情功能。
- 对于个人日记的删除管理功能。
- 对于全部公开日记的列表读取功能。
- 对于全部公开日记的读取详情功能。

注意：这里只是列出一些最基础的功能，因为它是一个展示性的入门级日记系统 Demo，而非一个真正完整的可用品。同样，对于这个 Demo 可以存在多个可优化的部分，读者可以发挥自己的想象力。

5.1.3 系统设计

系统的整体设计，是在需求分析和功能性分析后的又一个非常重要的环节，对于此环节的设计，直接影响到具体的编码工作，也是连接策划和程序的最关键的一环。一个良好的系统设计，可以深化对于用户的需求分析，减少很多编码时不必要的环境，使整个开发、产品的流程和架构清晰。

在这个环节中，需要假想多个应用性的场景，模拟用户进行产品的操作，在其中可以进行所有产品的流程模拟。为了使整个流程变得更加明确，可以通过一些流程图来实现整个系统的说明和设计。

注意：因为本书的主要内容是对于微信小程序的开发，而非是服务器端的开发，所以本章不会介绍服务器端的开发。对于小程序本身而言，使用 API 是开发小程序非常重要的一个环节，所以请读者利用已有的 API，或者自行学习后端开发功能。

在整体的系统设计中，小程序与服务器 API 的交互如图 5-1 所示。



图 5-1 小程序与 API 的交互

这样服务器端和客户端通过接口的交互方式（这是现阶段最常用的交互方式），通过后端开发的数据接口，客户端将一些需要处理或者做持久化的数据，以一定的形式和合适的验证方式，发送给后端服务器。

服务器接收到客户端发送的数据后，会将数据进行验证、存储，或其他处理，然后将结果返回给客户端，而客户端通过对此消息的解读，做出下一步的处理。

根据日记系统的功能设计，可以将其分为两个系统：

- 针对于用户的登录系统，也就是用户管理系统。
- 针对于日记的管理系统。

具体登录系统的流程，如图 5-2 所示。

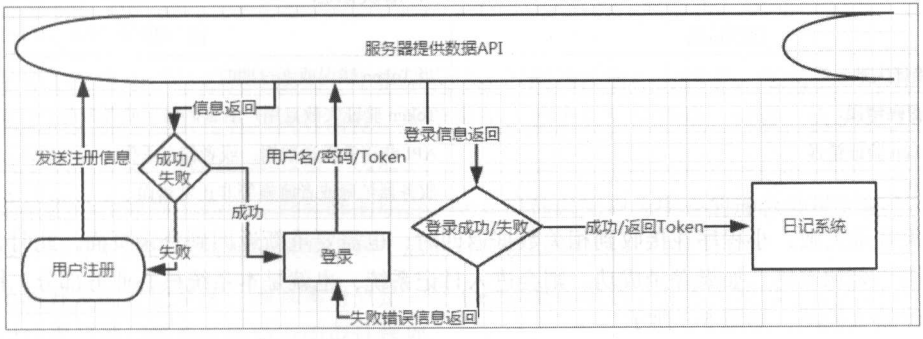


图 5-2 用户登录系统流程

注意：这里的流程图将整个后端服务器框架进行了透明化，不解析和介绍其实现的原理和内部结构。

根据图 5-2 可以看出，整个流程会从一个用户注册开始。也就是说，当用户第一次进入微信小程序时，首先要做的就是用户注册。在微信小程序端，所有的用户信息填写之后会将数据进行格式化，并发送给服务器端。

服务器端接收到相关的数据后，会对整个数据进行分析，并且判断此次数据是否真实有效，如

果数据有效则进行持久化存储，并返回注册成功的信息提示。如果信息不合法，则返回一个错误信息。返回的错误信息如表 5-1 所示。

表 5-1　返回的错误信息

错误信息	说　明
注册失败，账户重复	当账户信息重复注册时出现
开发 Token 验证错误	API 非正常发送数据，或者信息丢失
未知错误	服务器忙碌或者瘫痪等非正常错误

出现错误信息，说明此次用户注册失败，微信小程序端接收到此信息应当提示用户注册失败，并且返回到注册页面，让用户重新注册。如果用户注册成功，程序将会进行第二部分处理，也就是登录。

正常注册时，系统会返回一个有实效性的用户 Token，通过这个 Token 的记录（缓存），可以让用户实现在一段时间的持续登录，而不用每一次都要通过输入密码进行登录。

如果 Token 验证失败时，则会直接进行用户名和密码的验证，当验证成功登录后将返回用户信息和一个新的七天有效的 Token 值。

登录时的错误信息如表 5-2 所示。

表 5-2　登录时的错误信息

错误信息	说　明
Token 验证失败	当 Token 错误或者过期时
用户名密码错误	Token 验证失败且用户名密码错误或者为空
开发 Token 验证错误	API 非正常发送数据，或者信息丢失
未知错误	服务器忙碌或者瘫痪等非正常错误

如果登录失败，小程序在接收到相关数据返回时，也需要重新返回到登录页面，此时会要求用户输入用户名和密码。如果登录成功，则会进入日记系统，也就是本系统核心业务部分。针对日记系统设计的流程图，如图 5-3 所示。

日记系统依赖于用户系统登录后的用户 Token，通过此 Token 进行用户数据的查找和管理操作。用户登录后获取到 Token，将此 Token 发送给服务器的接口，会返回个人全部日记和所有人的公开日记，微信小程序的客户端会将所有的数据进行展示。

每一个数据内容，应该都是以列表形式进行展示的，而每个列表的项目都是可以单击的，通过单击此条目可以进入此条日记的详细页面。单击此条目时，微信小程序会将此条目的 id 发送给服务器端，在服务器端验证相关的操作后，会将此条日记的详细内容返回给微信小程序端，而小程序端接收到相关的数据后显示整篇日记。

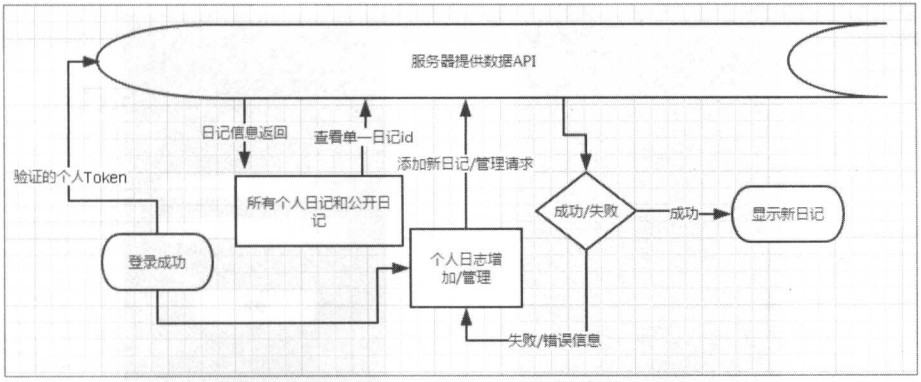


图 5-3 日记系统流程

不仅仅是查看，用户还可以对自己的日记进行增加和删除等管理操作。同样，也允许用户增加新的日记内容，在客户端中用户需要将相关的日记填写完毕，系统将数据打包发给服务器端，服务器端解析后将此数据进行持久化存储，并确定其所有者。当客户端接收到相关添加成功的信息时，会自动跳转回主页面，进行所有日记的更新和显示。

5.1.4 原型设计

如果是一个完整的产品开发流程，原型设计便是正式开发的第一步，需要在此环节中进行产品的界面、UI、UE 等设计。

产品原型是整个产品完成之前的一个框架设计。以网站注册作为例子，完成整个前期的交互设计流程图之后，就是原型开发的设计阶段。简单来说，是把页面的模块、元素、人机交互的形式，利用线框描述的方法，表达产品的各个功能。

本节的设计在原则上是交互设计师的职责，交互设计以用户为中心的理念会贯穿整个产品。利用交互设计师专业的眼光与经验激发出该产品的可用性。

本节将开发的日记系统，因其本身只是一个 Demo 类的项目，并不需要过于完善的 UE、UI 设计，所以这里只进行简单的原型页面设计规划。

注意：这里使用 Axure 进行简单的原型图绘制，因为这里只是让读者拥有原型设计的概念，所以绘制的原型图可能和具体产品有一些不同。

（1）用户进入页，即自动载入页（用于展示 logo 和名称，同时等待用户 Token 验证的返回），其原型图如图 5-4 所示。

（2）登录页面，两个输入框用于输入用户名和密码。下方有注册按钮，单击可以跳转至注册页面，其原型图如图 5-5 所示。

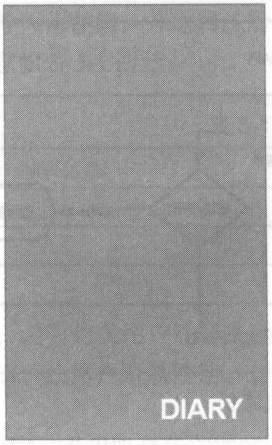


图 5-4　载入页面原型图



图 5-5　登录页面原型图

（3）注册页面，用户相关信息输入框和注册按钮，其原型图如图 5-6 所示。

（4）主页显示页面，显示所有公开的日记标题、简介，以及个人全部日记，以时间从近到远的顺序排列，其原型图如图 5-7 所示。

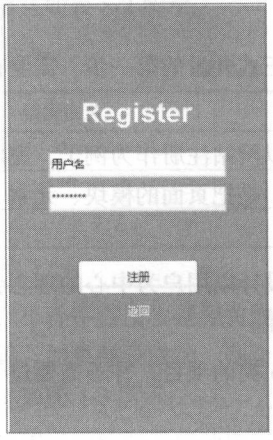


图 5-6　注册页面原型图

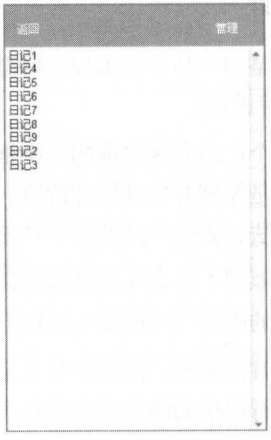


图 5-7　主页面原型图

（5）单篇日记显示页面，显示单篇日记内容、作者、发布时间等信息，其原型图如图 5-8 所示。

（6）日记添加页面，日记的相关内容文本框以及提交按钮，其原型图如图 5-9 所示。

（7）个人日记管理页面，显示所有的日志列表，并且在列表后增加一个管理按钮选项，这里只做一个删除选项，其原型图如图 5-10 所示。

这样，所有页面的原型图全部设计完毕，下一步，便进入了正式编码开发的部分。

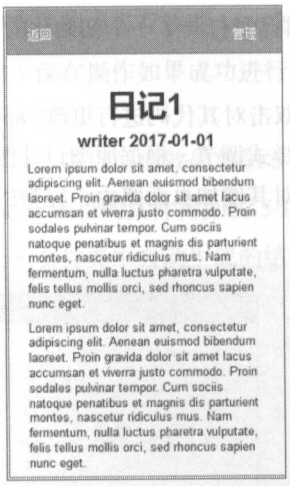


图 5-8 单篇日记页面原型图

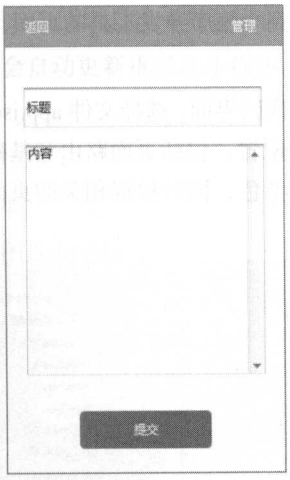


图 5-9 添加页面原型图

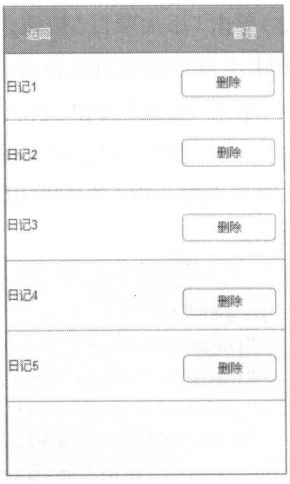


图 5-10 管理页面原型图

5.2 具体编码

从本节开始正式进入编码部分，我们会详细介绍完成整个项目的全部内容。

注意：本节中可能会涉及下一章的知识点，读者可以自行进行查阅。

5.2.1 项目初始化

（1）打开安装好的微信小程序开发工具，开始进行整个项目的开发工作。首先，需要新建项目，如图 5-11 所示。

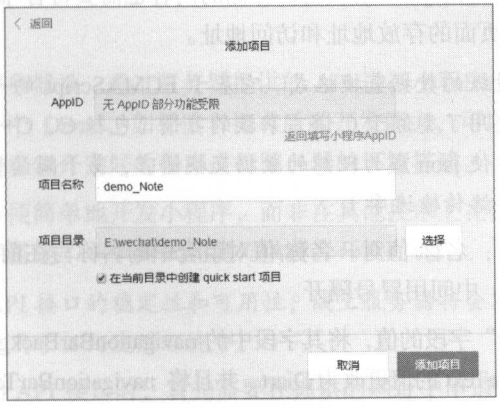


图 5-11 新建项目

（2）单击“添加项目”按钮，进入到小程序开发工具的页面，此时已经打开了一个初始化的 quick start 项目。

（3）单击“编辑”按钮，进入编辑代码界面，选择文件 app.json，双击对其代码进行更改。app.json 内部的代码主要的作用是对小程序的配置，包括页面路由和基础的样式调节。根据原型设计图中的颜色，需要调整 color 为#9999ff 的淡紫色，同时根据相关的页面，对其增加页面路由，完整的代码如下所示。

```
{
  "pages":[
    "pages/index/index",
    "pages/index/home",
    "pages/index/diary",
    "pages/index/add",
    "pages/index/manager",
    "pages/logs/logs",
    "pages/user/index",
    "pages/user/login",
    "pages/user/register"
  ],
  "window":{
    "backgroundTextStyle":"light",
    "navigationBarBackgroundColor": "#9999ff",
    "navigationBarTitleText": "Diary",
    "navigationBarTextStyle":"write"
  }
}
```

对于该文件来说，其本身是 json 格式的，存储的内容也需要符合 json 形式的字符串，其中的 pages 字段的值，就是所有页面的存放地址和访问地址。

注意：json 是一种轻量级的数据交换格式。它基于 ECMAScript 的一个子集，采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C、C++、C#、Java、JavaScript、Perl、Python 等）。这些特性使 json 成为理想的数据交换语言，易于阅读和编写，同时也易于机器解析和生成（一般用于提升网络传输速率）。

json 数据的书写格式是：名称/值对。名称/值对组合中的名称写在前面（在双引号中），值对写在后面（同样在双引号中），中间用冒号隔开。

根据需求更新“window”字段的值，将其字段中的 navigationBarBackgroundColor 对应的值更改为 #9999ff，将 navigationBarTitleText 的值更改为 Diart，并且将 navigationBarTextStyle 的值改为白色 write。

（4）根据需求，设计 9 个不同的页面用来显示内容，与日记有关的内容存放在 index 文件夹下，

与用户有关的内容存放在 user 文件夹下，使用 Ctrl+s 组合键保存。

（5）保存操作如果成功进行，将会自动更新重启此小程序，并且会自动在项目中新建相关的文件夹和文件。

此时，重启后的效果如图 5-12 所示。

Index 项目的文件树如图 5-13 所示。

user 文件夹中自动生成的结构如图 5-14 所示。

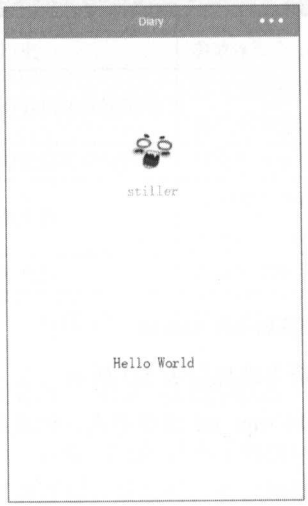


图 5-12 重启后的效果



图 5-13 index 项目的文件树



图 5-14 user 文件夹中自动生成的结构

这样，一个小程序的结构框架搭建完毕，剩下的操作就是具体的编码了。

5.2.2 测试环境搭建

小程序的开发会用到服务器端，通过服务器提供的 API 进行小程序的开发操作。本章是作者配合服务器后端已经写好的一个 Demo，如果读者拥有后端开发能力，可以自行开发相关的 API，或者下载相关的程序包，使用已经开发好的程序进行服务器的环境搭建。

为了让读者能够更加方便简单地开发小程序，而非在其他技术上花费大量的时间和精力，所以作者在公网上的云服务器中搭建了一个测试环境，供读者开发时调用。

注意：为了保证相关 API 接口的稳定性和可用性，线上服务器将会不定期进行数据删除和清洗操作，所以请不要将本地址用于任何使用环境中。

使用线上服务器提供的 API 接口时，首先需要在提供的网址中申请开发者 Token，用于接口的验证。

（1）打开浏览器，输入地址 http://114.215.29.122:44/，按 Enter 键进入主页，如图 5-15 所示。

注意：申请的基础密钥为 DsmMdsMMY1Gdd。

（2）在此页面中输入正确的密码和邮箱。单击“提交获取 Token”按钮就能获取一个 Token。每个开发者的 Token 都是唯一的一串字符，请开发者妥善保管。本页面提供 Token 找回功能，只要在此页面输入相同的信息，就会返回一个 Token 值，这个值是验证 API 请求时的唯一密钥，其效果如图 5-16 所示。

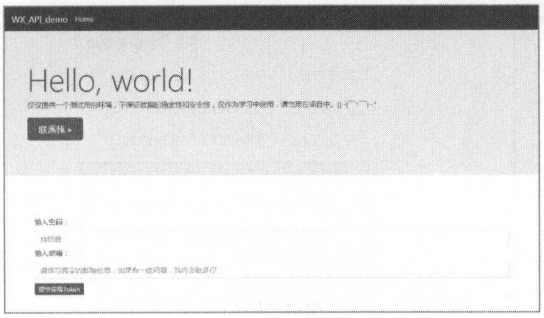


图 5-15　申请开发者 Token

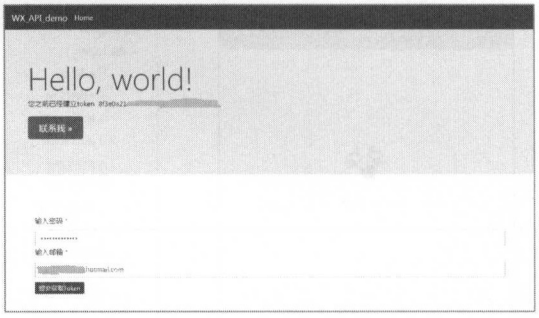


图 5-16　获取 API 的密钥

（3）对于此服务器提供的相关接口，基于上述项目的原型设计，请求地址如表 5-3 所示。

注意：以上所有接口地址线上测试环境均是 http://114.215.29.122:44/+api_url 这种形式，但是所有的请求均是 POST 方式。

两种最常用的 HTTP 方法是：GET 和 POST。

- GET：从指定的资源请求数据。
- POST：向指定的资源提交要被处理的数据。

GET 和 POST 的不同点如表 5-4 所示。

表 5-3　API的地址和说明

接口地址	说　明
api/register	用户注册地址
api/login	用户登录获取用户 Token 地址
api/write	用户增加新日记地址
api/read	用户读取所有个人日记地址
api/list	用户读取所有个人日记列表
api/delete	用户删除操作地址
api/plist	用户读取公开列表地址
api/pread	用户读取公开日记地址

表 5-4 GET和POST的不同点

项 目	GET 方法	POST 方法
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）
书签	可收藏为书签	不可收藏为书签
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded 或 multipart/form-data，为二进制数据使用多重编码
历史	参数保留在浏览器历史中	参数不会保存在浏览器历史中
对数据长度的限制	当发送数据时，GET 向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）	无限制（理论）
对数据类型的限制	只允许 ASCII 字符	没有限制，也允许二进制数据
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 中的一部分	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 Web 服务器日志中
可见性	数据在 URL 中对所有人都是可见的	数据不会显示在 URL 中

具体的地址请求参数会在下一节详细介绍。

5.2.3 用户系统编码

在编码之前必须了解微信小程序的发起请求 API：使用 wx.request(OBJECT)进行一个 HTTPS 请求的发起。发起时，可以增加一些请求参数，通过对参数的配置，可以访问不同的数据接口。具体的参数说明如表 5-5 所示。

表 5-5 参数说明

参 数 名	类 型	必 填	说 明
url	String	是	开发者服务器接口地址
data	Object、String	否	请求的参数
header	Object	否	设置请求的 header，header 中不能设置 Referer
method	String	否	默认为 GET，有效值：OPTIONS、GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT
dataType	String	否	默认为 json。如果设置了 dataType 为 json，则会尝试对响应的数据做一次 JSON.parse
success	Function	否	收到开发者服务器成功返回的回调函数，res={data:'开发者服务器返回的内容'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

说明：表 5-5 中的内容和实例下一章会详细介绍。

下面开始功能编码。

1. 开屏页

首先实现的页面是一个展示页面。按照之前的原型图和系统设计，需要在这个页面中显示小程序的一个文字 logo，并且停留 3~5 秒后自动跳转出此页面，即仿照常见 App 制作一个开屏页（读者可以理解为孔雀开屏）。

这个页面使用系统默认的主页 index/index 路由来编写。首先打开 index.wxml 文件，写一个简单的模板。

```
<!--index.wxml-->
<view class="main">
  <view class="main_logo">
    <text class="main_text">DIARY</text>
  </view>
</view>
```

此页面文件主要展示了一个大写的文字 logo，并且可以对样式进行相应的调节，使其和原型设计图保持一致。

页面样式文件的具体代码如下：

```
/**index.wxss**/
.main {
  background: #9999ff;
  width: 100%;
  height: 598px;
}
.main_logo{
  padding-right: 10%;
  padding-top: 550px;
}
.main_text{
  font-size: 30px;
  color: #ffffff;
  float: right;
}
```

这样，一个开屏页就制作完毕了，其显示效果如图 5-17 所示。

注意：这里为了方便开发，一些参数采用了绝对高度等设置方式，这会导致在部分设备上出现不能全屏的问题。本程序开发时使用的设备为 Galaxy S5，Wi-Fi 环境。

上述代码仅仅调整了样式，并不是最重要的一部分，真正要实现的是自动页面跳转。这里可以

使用简单的 JS 函数 setTimeout()进行延迟跳转，这段代码需要写在 index 页面的 onload 状态下。index.js 完整的代码如下：

```
//index.js
//获取应用实例
var app = getApp()
Page({
  data: {
  },
  onLoad: function () {
    setTimeout(function(){
      wx.redirectTo({
        url: '../user/index'
      })
    },3000
    )
  }
})
```

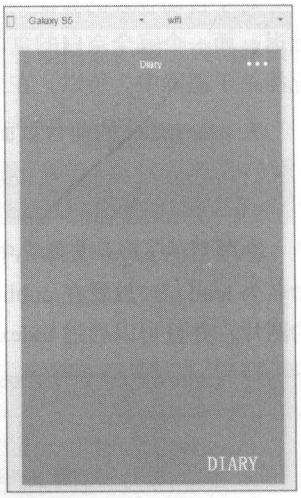


图 5-17 开屏页

使用 Ctrl+s 组合键保存代码，小程序编辑器会自动重启此小程序，在 index 页面停留 3 秒后进入 user/index 页面。

2. 验证用户的登录历史

在 user/index 页面验证用户的登录历史，如果把从未登录或者登录已经过期的用户 Token 发送给服务器，服务器返回成功后会跳转主页面，如果返回登录失败，则会返回到用户名和密码的登录页面。

这里请求的地址是 http://114.215.29.122:44/api/login。其访问方式为 POST，可以将 Token、用户 Token、用户名、密码等作为请求参数，一起发送给后端服务器，而服务器验证其有效性和正确性后会返回对应的值。

具体的参数如表 5-6 所示。

表 5-6 登录请求参数

请求参数	类 型	说 明
token	String	开发者验证 Token
username	String	使用者用户名
password	String	使用者密码
usertoken	String	使用者上次登录 Token

现阶段，因为本例暂时没有任何用户和登录信息，除非手动在数据库中增加数据，否则无法进行登录。但不用担心，先将此登录阶段的功能做出一部分，在完成用户注册和登录功能后再完整编写。

对于自动登录的验证页面，因为考虑到用户不同的网络环境，在弱网络的情况下可能会出现出延迟，而一个完全空白的页面会让用户怀疑此小程序的可用性，所以尽量把可能造成用户流失的情况扼杀于襁褓中。所以，在未收到回复时显示小程序中的一个 gif 图片。

在 user/index 的前台页面编写如下代码：

```
<!--pages/user/index.wxml-->
<image class='load_pic' src="../public/img/load.gif"></image>
```

在项目的 pages 文件夹中建立一个用于放置静态文件的文件夹 public，制作一张 gif 动态载入图，命名为 load.gif，放置在 public 文件夹下的 img 图片文件夹中。这样在此页面中就成功引入了 loading 的图片，并且可以通过 index.wxss 调整样式，让其位于页面中间。

```
/* pages/user/index.wxss */
.load_pic{
    padding-top: 60%;
    padding-left: 35%;
    width: 140px;
    height: 100px;
}
```

这里需要对服务器的登录 API 进行请求，因为没有缓存的数据，所以暂时先不发送相关参数。使用 wx.request 进行请求的发送，其具体的参数和解析请见其他章节。

```
//pages/user/index.js
Page({
  data:{},
  onLoad:function(options){
    //页面初始化 options 为页面跳转所带来的参数
    wx.request({
      url:'http://114.215.29.122:44/api/login',//仅为示例，并非真实的接口地址
      data: {},
      method:'POST',
      header: {
        'content-type': 'application/json'
      },
      success: function(res) {
        console.log(res.data)
      }
    })
  }
})
```

这里使用 wx.request()发送请求，请求方式为 POST，请求地址为 http://114.215.29.122:44/api/login

测试地址，暂时未发送任何参数。同时，将结果打印在控制台中。

使用 Ctrl+s 组合键保存代码，并且单击“调试”按钮，查看控制台打印信息。调试台会将服务器返回的值打印出来。打印结果如图 5-18 所示。

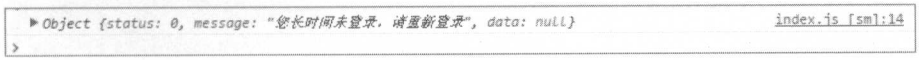


图 5-18 控制台打印返回值

这类 json 的返回值，证明发送请求后不成功的状态，status 为 0 时，证明出现了一个错误。具体的通用返回值说明如表 5-7 所示。

表 5-7 服务器通用返回值说明

返 回 值	类 型	说 明
status	Int	当返回值为 1 时，请求成功，为 0 时不成功
message	String	当返回值为 0 时会出现相关的错误提示信息，而返回值为 1 时，此信息一般为 null 的状态
data	String	当返回值为 1 时出现的返回值，如果当返回的状态值为 0 时，此 data 为 null 的状态

根据表 5-7 所示，可以将请求完毕的跳转逻辑完善，当返回值为 1 时，跳转到 index/home 日记的主页中，当返回值为 0 时，跳转到登录的页面路由 user/login。所以一个完整的 index.js 的代码为：

```
//pages/user/index.js
Page({
  data:{},
  onLoad:function(options){
    //页面初始化 options 为页面跳转所带来的参数
    wx.request({
      url: 'http://114.215.29.122:44/api/login',//仅为示例，并非真实的接口地址
      data: {},
      method:'POST',
      header: {
        'content-type': 'application/json'
      },
      success: function(res) {
        console.log(res.data)
        if(res.data.status==1){
          wx.redirectTo({
            url: '../index/home'
          })
        }else{
          wx.redirectTo({
            url: './login'
```

```
            })
        }
    }
})
}
})
```

上述代码通过对接口请求数据的解析，获得返回的接口值，并通过 wx.redirectTo()函数进行相关页面的跳转。

注意：wx.redirectTo()的参数和详细解析，见其他章节。

使用 Ctrl+s 组合键保存代码，因为此时未传递任何参数，所以小程序重启后会自动跳转到登录页面。

也可以通过调整当前测试的网络环境，来测试此页面在弱网络或无网络状态下的 UE 表现。网络状态为 none 时的界面如图 5-19 所示。这个状态比完全的白屏情况好很多。

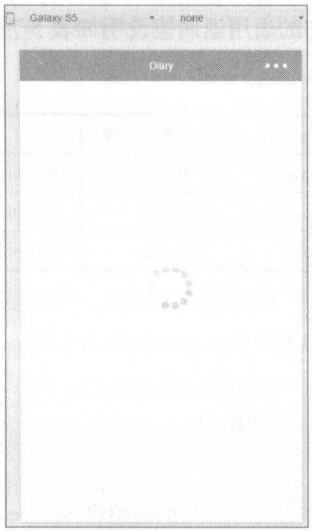


图 5-19　网络状态为 none 时的界面

注意：在更改网络状态时，更改后的网络状态重启应用必须单击左下方的“重启”按钮，因为未对代码进行更改，所以此时 Ctrl+s 组合键是无效的。

3. 用户的登录界面

按照之前的系统设计和原型图，这个界面上需要一个表单元素，包含可以输入用户名和密码的两个文本框，还应当有“登录”按钮和单击会跳转至注册页面的“注册”按钮。

完整的登录页面代码如下所示。

```
<!--pages/user/login.wxml-->
<view class="main">
    <form bindsubmit="formSubmit" bindreset="formReset">
        <view class="main_text">
            <text class="login_word">LOGIN</text>
            <input name="username" class="input" placeholder="用户名"/>
            <input name="password" class="input" password placeholder="密码"/>
            <view>
                <button formType="submit" class="login_btn" size="mini">登录
</button>
            </view>
            <view>
```

```
                <button bindtap="re_Register" class="login_btn" size="mini">
注册</button>
            </view>
        </view>
    </form>
</view>
```

表单元素需要包裹在<input>的外部，同时对文本框元素的 name 赋值，分别是 username 和 password，这样在发送请求时可以将其值取出，发送给服务器端进行登录操作。

“注册”按钮采用了事件绑定的方式，绑定了 JS 中的一个 re_Register()方法。当然，依旧需要对页面的样式进行相应的调整，其 login.wxss 的完整代码为：

```
/* pages/user/login.wxss */
.main{
    background: #9999ff;
    width: 100%;
    height:598px;
}
.main_text{
    padding-top: 80px;
    padding-left: 40%;
}
.login_word{
    color: #ffffff;
    font-size: 30px;
}
.input{
    background: #ffffff;
    margin-left: -20%;
    margin-top: 10%;
    width: 80%;
}
.login_btn{
    margin-top:10%;
    margin-left: 3%;
}
```

然后是 JS 处理的逻辑页面，主要是包含了两个按钮的单击事件，单击“注册”按钮会直接跳转至注册页面，而单击“登录”按钮会通过 wx.request()方法给服务器发送所有表单中的数据。

注意：此请求和通过 Token 请求 API 地址都是一致的，所以请求相同地址，发送不同的请求参数即可。

不同于自动登录需要发送的 user_token 和验证 Token，登录操作需要发送用户名和密码，以及验证 Token，才可以完成此功能。

所有来自客户端的请求参数中都关联一个开发者的 Token 值，这样可以防止一些未知来源的恶意 API 请求，也可以一定程度上检查传输过程中是否存在信息的丢失。所以对于开发者 Token 这样常用的值，应当使用一个常量进行存储，微信小程序为开发者提供了这样的全局变量。

这个全局变量的定义在 app.js 中，通过修改 globalData 可以添加一个全局值，本例代码添加如下：

```
//记录开发者 Token 等
globalData:{
  userInfo:{
    dev_token:"8f3e0a21********************"
  }
}
```

在开发者需要全局值的时候，只需要获取 globalData 属性就可以完成。本例取得 Token 的代码如下：

```
getApp().globalData.userInfo.dev_token
```

一个用户的登录请求需要发送三个参数进行验证，分别是使用者的用户名和密码，以及开发者的验证 Token。

登录页面本身需要实现的基础逻辑是：将用户填写在表单文本框中的值发送给服务器，如果登录成功，服务器端返回一个验证的用户 Token，这个 Token 值的意义在于，实现 user/index 页面的自动登录，其有效值为 7 天。当用户输入错的用户名或密码时，显示相关的错误，并且不跳出此登录页面，而是提示用户重新输入。

完整的代码如下：

```
//pages/user/login.js
Page({
    re_Register: function(e) {
      wx.navigateTo({
        url: './register',
      })
    },
    formSubmit: function(e) {
      //当单击了“submit”按钮后发出新的请求
      wx.request({
        url:'http://114.215.29.122:44/api/login',//仅为示例,并非真实的接口地址
        data: {
          'token':getApp().globalData.userInfo.dev_token,
```

```
        'username':e.detail.value.username,
        'password':e.detail.value.password,
      },
      method:'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function(res) {
        console.log(res.data)
        if(res.data.status==1){
          //存入缓存中
          wx.setStorage({
            key:"username",
            data:JSON.parse(res.data.data).username
          })
          wx.setStorage({
            key:"usertoken",
            data:JSON.parse(res.data.data).usertoken
          })
          wx.redirectTo({
            url: '../index/home'
          })
        }else{
          wx.showToast({
            title: res.data.message,
            icon: 'loading',
            duration: 2000
          })
        }
      }
    })
  },
  data:{},
})
```

错误的提示使用了一个弹出层，通过 wx.showToast()方法显示一个两秒的消息。具体的参数和说明会在下一章做详细介绍。

当验证成功时，服务器端会返回一些已经记录的参数和用户的 user_token，以供下一次的自动登录，这里的处理方式是使用微信小程序自带的缓存，使用 wx.setStorage()方法。缓存包含一个键值对。使用 console.log(res.data) 打印返回值，服务器返回的结果为一个 json 串，如图 5-20 所示。

登录成功后，该接口返回两个数据，代表用户登录成功，具体的数据和说明如表 5-8 所示。



图 5-20　登录返回

表 5-8　用户登录返回值

返回数据名称	类　型	说　明
username	Sting	用户登录的用户名
usertoken	String	用户登录时自动生成的 Token，此 Token 用于记录用户的登录状态与实现自动登录功能，其有效期为 7 天

这里需要缓存两个值，这样可以在某些需要显示用户名的地方直接使用，一些页面请求需要 usertoken 进行验证。

注意：对于返回值而言，其返回一个 json 对象，对于 JavaScript 自身而言是可以直接解析的，但是对于其 data 属性，服务器提供的只是一个 String 字符串（虽然此字符串也符合 json 格式），所以开发者需要使用 JSON.parse(res.data.data)这样的方式将 String 转换成 JavaScript 可以直接解析的 json 对象。

使用 Ctrl+s 组合键进行保存，微信小程序会自动重启，再自动跳转至登录界面，如图 5-21 所示。

当输出错误的用户名或者密码时，程序将不会跳出此界面，并且会出现一个弹出层，提示错误信息，如图 5-22 所示。

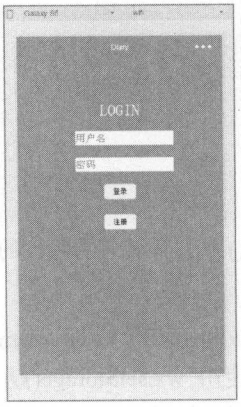


图 5-21　登录界面

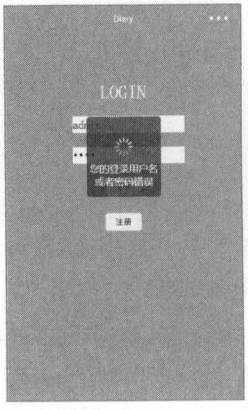


图 5-22　信息错误

输入正确的用户名和密码后，微信小程序将会自动跳转至 index/home 页面中。登录成功后，单击“调试”按钮进入调试界面，再单击调试工具的 Storage 选项查看此程序的缓存，可以看到登录用户的用户名称和 Token 值，如图 5-23 所示。

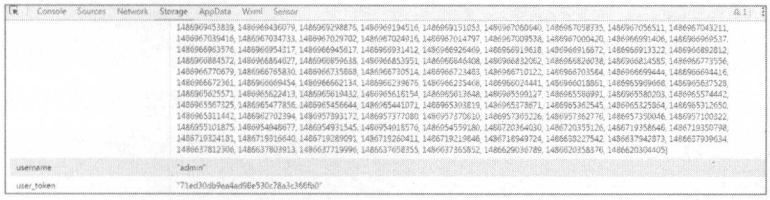


图 5-23 微信小程序的缓存

注意：为了方便读者调试，测试服务器提供了一个账号和密码均为 admin 的测试账户。

4. 自动登录

既然已经获取到了用户的 Token，下一步要先完成为用户自动登录设计的发送数据功能。

单击“编辑”按钮，进入编辑页面，将 user/index 路由中的 JS 文件进行更改，从缓存中读取其 usertoken 值，结合开发者的 Token，将其发送给服务器端，检测是否登录成功。其完整代码如下所示。

```
//pages/user/index.js
Page({
  data: {},
  onLoad: function (options) {
    //页面初始化 options 为页面跳转所带来的参数
    wx.request({
      url: 'http://114.215.29.122:44/api/login',//仅为示例，并非真实的接口地址
      data: {
        'token': getApp().globalData.userInfo.dev_token,
        'usertoken': wx.getStorageSync('usertoken')
      },
      method: 'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        //console.log(wx.getStorageSync('usertoken'))
        console.log(res.data)
        if (res.data.status == 1) {
          wx.redirectTo({
            url: '../index/home'
          })
```

```
        } else {
          wx.showToast({
            title: res.data.message,
            icon: 'loading',
            duration: 1000,
            success: function (res) {
              wx.redirectTo({
                url: './login'
              })
            }
          })
        }
      }
    })
  }
})
```

注意：因为服务器端的要求，发送数据的头部信息必须使用 Unicode 形式，而不能使用默认的 json 形式。同理，该应用的所有请求均如此，即需要将 header 改为'content-type': 'application/x-www-form-urlencoded'。

在由 Token 实现的自动登录中，服务器端是不会更新用户 Token 值的，所以无须将返回的 Token 进行缓存。登录成功后，页面也会自动跳转至 index/home 页面。

5. 用户的注册

单击左下方的“缓存”按钮，清除当前所有的缓存，这时可以看到调试窗口的 Storage 均已被清除。重启当前小程序。由于缓存被清除，所以小程序不会自动登录成功而进入 index/home，而是会跳转至登录页面，此时单击“注册”按钮，会跳转至 user/register 页面。

此链接的跳转，使用了保留登录页面的方式，而非直接在登录页中跳转，所以左上方会自动出现“返回”按钮，单击此按钮会返回至登录页面。完整的 register.wxml 代码如下所示。

```
<!--pages/user/register.wxml-->
<view class="main">
    <form bindsubmit="formSubmit" bindreset="formReset">
        <view class="main_text">
            <text class="login_word">REGISTER</text>
            <input name="username" class="input" placeholder="用户名"/>
            <input name="password" class="input" password placeholder="密码"/>
            <input name="re_password" class="input" password placeholder="
重新输入密码"/>
            <view>
```

```
                <button formType="submit" class="login_btn" size="mini">注册
</button>
            </view>
        </view>
    </form>
</view>
```

略微修改登录页面就可以获得新的注册页面，其样式文件和登录页面是一致的。

```
/* pages/user/register.wxss */
.main{
    background: #9999ff;
    width: 100%;
    height:598px;
}
.main_text{
    padding-top: 80px;
    padding-left: 40%;
}
.login_word{
    color: #ffffff;
    font-size: 30px;
}
.input{
    background: #ffffff;
    margin-left: -20%;
    margin-top: 10%;
    width: 80%;
}
.login_btn{
    margin-top:10%;
    margin-left: 3%;
}
```

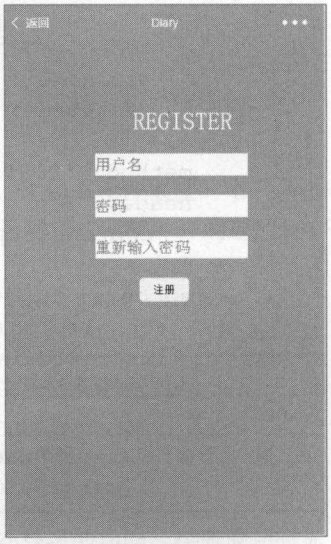


图 5-24 注册页面

注册页面显示效果如图 5-24 所示。

注册请求 API 地址为 http://114.215.29.122:44/api/register，其接收的请求参数为用户的用户名和密码。注册请求参数如表 5-9 所示。

表 5-9 注册请求参数

请求参数名	类 型	说 明
username	String	用户名
password	String	密码
token	String	开发者验证 Token

两次输入密码需要在客户端验证是否一致。当不一致时不发送请求，并且给出一个弹出层的提示信息。完整的 JavaScript 代码如下所示。

```
//pages/user/register.js
Page({
  data: {},
  formSubmit: function (e) {
    //对于是否两次密码输入一致进行判断
    if (e.detail.value.password == e.detail.value.re_password) {
      //当单击了 submit 按钮后发出新的请求
      wx.request({
        url: 'http://114.215.29.122:44/api/register', //仅为示例，并非真实的接口地址
        data: {
          'token': getApp().globalData.userInfo.dev_token,
          'username': e.detail.value.username,
          'password': e.detail.value.password,
        },
        method: 'POST',
        header: {
          'content-type': 'application/x-www-form-urlencoded'
        },
        success: function (res) {
          console.log(res.data)
          if (res.data.status == 1) {
            //存入缓存中
            wx.setStorage({
              key: "username",
              data: JSON.parse(res.data.data).username
            })
            wx.setStorage({
              key: "usertoken",
              data: JSON.parse(res.data.data).usertoken
            })
            wx.showToast({
              title: res.data.message+",请返回登录。",
              icon: 'success',
              duration: 2000
            })
          } else {
            wx.showToast({
              title: res.data.message,
```

```
            icon: 'loading',
            duration: 2000
          })
        }
      }
    })
  } else {
    wx.showToast({
      title: "两次输入密码不一致，请检查",
      icon: 'loading',
      duration: 2000
    })
  }

 },
})
```

通过对单击表单提交事件的检测，小程序端会判断两次输入密码是否一致，如果两次输入不一致，则会直接提示，如图 5-25 所示。

同样，对于用户的注册，服务器端也提供了一些简单的验证信息，比如不允许重复的用户名进行注册等功能。信息的返回值依旧符合 json 标准。当注册成功时，会返回和登录时一样的返回值，如表 5-10 所示。

表 5-10 用户登录返回值

返回数据名称	类 型	说 明
username	Sting	用户登录的用户名
usertoken	String	用户登录时自动生成的 Token，此 Token 用于记录用户的登录状态与实现自动登录功能，其有效期为 7 天

注册成功时自动保存用户注册的用户名和 Token 值，提供下次自动登录的功能。注册成功时显示效果如图 5-26 所示。

注意：这里只是简单的注册示例，并未对密码和传输本身进行加密，这在一个正式的程序中是不应该的。

单击左上方的“返回”按钮，返回登录页面，输入注册时的用户名和密码，即可登录成功。直接重启小程序本身，也可以自动登录成功。

本节的用户系统编码过程已经完成。下一节将开发日记系统的主要功能。

注意：微信小程序并不需要使用者手动建立用户名和密码，可以通过微信的自动登录实现登录操作，本节主要的目的是为读者展现一个开发的流程，并且对此进行分析。

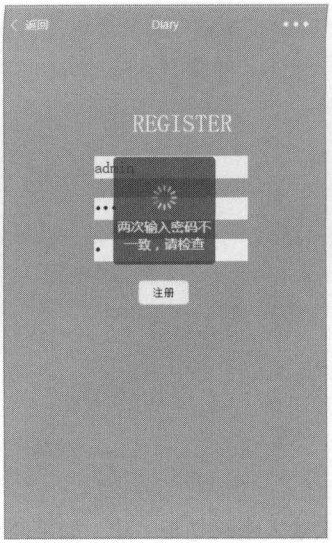


图 5-25　密码一致性验证

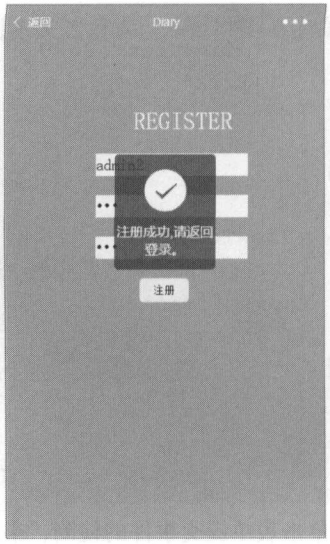


图 5-26　注册成功

5.2.4　日记系统编码

上一节完成了一个基础用户管理的雏形，本节开始日记系统最主要的部分。

1. 主页

首先是用户登录后的主页部分，在 index/home 路由下的主页，此页面显示所有用户的日记信息，通过列表的形式显示所有的文章类型。主要是显示一篇文章的标题、作者和发布日期等信息，并且对于其显示是可以单击的，单击进入新的页面，此页面会显示该文章的详情。

对于显示多条数据，使用 for 循环标签的形式进行遍历显示，使用<navigator>标签来包裹选择项目。其链接的 URL 地址中需要将此文章的 id 传入日记文章的详情页中，由 index/diary 拿到请求的参数后，向服务器再次发出请求，才可以获取详情。

wxml 文件的具体代码如下所示。

```
<!--pages/index/home.wxml-->
<view class="main">
  <view class="list" wx:for="{{items}}" wx:key="id">
    <navigator url="diary?id={{item.id}}" hover-class="navigator-hover">
      <text class="article_title">{{item.title}}</text>
      <text class="article_username">作者: {{item.username}}</text>
      <view>
        <text class="article_date">日期: {{item.date}}</text>
```

```
        </view>
    </navigator>
  </view>
</view>
```

当然，需要对其样式进行调整，为一个<view>标签设置一个底部边框，用于分割每一个不同的文章链接。wxss 文件的具体代码如下所示。

```
/* pages/index/diary.wxss */
.article_title{
    font-size: 28px;
    text-align:center;
    padding-top:20px;
}
.article_date{
    font-size: 12px;
    text-align:center;
    padding-top:10px;
    color: gray;
}
.article_username{
    font-size: 12px;
    text-align:center;
    color: gray;
}
.article_text{
    font-size: 18px;
    text-align:center;
    padding-top:20px;
}
```

而对于此页面，需要在载入的时候进行接口请求。测试服务器的接口地址为 http://114.215.29.122:44/api/plist，请求需要发送的两个相参数，一个是开发者 Token，另一个是用户 Token。

服务器在接收到相关的请求后，也会返回相关的值，对于 data 具体的返回值和说明如表 5-11 所示。

表 5-11 文章列表请求返回

data 中字段名称	类 型	说 明
id	Int	文章 id 唯一
title	String	文章标题
date	String	文章发布日期
uid	Int	文章作者 id
username	String	文章作者名称

这里使用了一个在全局变量临时存储页面载入时获取到的服务器中的返回值，通过 this.setData() 方法在页面绘制结束后在页面中重新绑定此值。完整的 JS 文件代码如下所示。

```
//pages/index/home.js
var temp_data;
Page({
  data: {
    items:[]
  },
  onLoad: function (e) {
    //页面初始化后发出新的请求
    wx.request({
      url: 'http://114.215.29.122:44/api/plist',//仅为示例，并非真实的接口地址
      data: {
        'token': getApp().globalData.userInfo.dev_token,
        'usertoken': wx.getStorageSync('usertoken')
      },
      method: 'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        console.log(res.data)
        if (res.data.status == 1) {
          // console.log(JSON.parse(res.data.data))
          temp_data = JSON.parse(res.data.data)
        } else {
          wx.showToast({
            title: res.data.message,
            icon: 'loading',
            duration: 2000
          })
        }
      }
    })
  },
  onReady: function () {
    //页面渲染完成
    this.setData({
      items:temp_data
    })
  },
})
```

通过数据的获取和绑定，即可实现一个文章列表，此时的前台页面，通过单击会自动跳转至 index/diary 页面，并且会发送一个唯一的文章 id 作为此篇文章的识别符，其主页显示效果如图 5-27 所示。

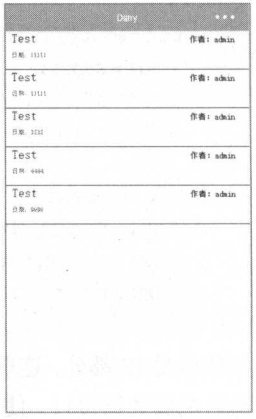


图 5-27 日记主页显示效果

2. 文章详情页

文章详情页是文章列表页单击具体链接时跳转的页面，路由为 index/diary。其本质与主页差不多，但是在此接口上获取到的还有内容部分，对于一篇文章而言，其内容部分可以是非常长的，在此页面上只需要显示一篇文章的标题、作者和内容。所以 wxml 的内容为：

```
<!--pages/index/diary.wxml-->
<view class="article_title">
  <text>{{article.title}}</text>
</view>
<view class="article_date">
  <text>发布日期：{{article.date}}</text>
</view>
<view class="article_username">
  <text>作者：{{article.username}}</text>
</view>
<view class="article_text">
  <text>{{article.text}}</text>
</view>
```

此页面的值只需要绑定一个变量，而不需要使用标签的循环方式，当然，在 JS 中处理数据时也需要将其放在一个变量中。

为了让显示更加人性化，需要对页面的样式进行调整，让文字显示为居中，行与行之间存在间隔。wxss 完整代码如下所示。

```
/* pages/index/diary.wxss */
.article_title{
    font-size: 28px;
    text-align:center;
    padding-top:20px;
}
.article_date{
    font-size: 12px;
    text-align:center;
    padding-top:10px;
    color: gray;
```

```
}
.article_username{
    font-size: 12px;
    text-align:center;
    color: gray;
}
.article_text{
    font-size: 18px;
    text-align:center;
    padding-top:20px;
}
```

然后是 JS 部分，这里的基本思路是先获得由主页发送至此页面的文章识别唯一 id，并且将其发送给后端服务器接口。使用 onLoad: function (options) {}方法中的 option 进行参数的获取，在小程序页面打开时，options 即为页面跳转所带来的参数。通过控制台的打印方法可以知道，其本身也是一个 JS 对象，所以对于唯一参数的获取，直接使用 options.id 即可。

获取文章接口的测试地址为 http://114.215.29.122:44/api/pread。请求存在三个参数，如表 5-12 所示。

表 5-12　文章请求参数

请求参数名	类　型	说　明
usertoken	String	用户登录后的 Token 值用于验证用户
articleid	Int	唯一的文章 id 来源于列表页的请求参数
token	String	开发者验证 Token

将这三个参数发送至服务器的接口时，服务器会返回该篇日记的相关内容，包含了其文章的返回值，如表 5-13 所示。

表 5-13　详细日记返回data数据的说明

data 中字段名称	类　型	说　明
id	Int	文章 id 唯一
title	String	文章标题
date	String	文章发布日期
uid	Int	文章作者 id
username	String	文章作者名称
text	String	文章内容
status	Int	文章是否公开，其值为 1 时公开，为 0 时则不公开

注意：这里返回的 data 数据本身是和列表一致的，返回一个数组，其中包含着一个 object，所

以赋值时需要使用 data[0]进行取值。

```
//pages/index/diary.js
var temp_data;
Page({
  data: {
    article: {}
  },
  onLoad: function (options) {
    //页面初始化 options 为页面跳转所带来的参数
    //console.log(options)
    wx.request({
      url: 'http://114.215.29.122:44/api/pread',//仅为示例，并非真实的接口地址
      data: {
        'token': getApp().globalData.userInfo.dev_token,
        'usertoken': wx.getStorageSync('usertoken'),
        'articleid': options.id
      },
      method: 'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        if (res.data.status == 1) {
          console.log(JSON.parse(res.data.data))
          temp_data = JSON.parse(res.data.data)
        } else {
          wx.showToast({
            title: res.data.message,
            icon: 'loading',
            duration: 2000
          })
        }
      }

    })
  },
  onReady: function () {
    //页面渲染完成
    this.setData({
      article: temp_data[0]
    })
```

```
  },
})
```

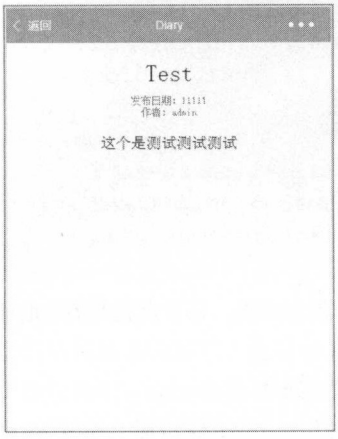


图 5-28　日记具体页面

通过 wx.request()发送请求，成功获得相关的返回参数后，重新设定 article 的值，显示相关的前台页面。具体效果如图 5-28 所示。

这样就完成了对于单一日记的详情查看功能，这里文章列表前台的页面使用了保留前页的方法，所以单击左上角的“返回”按钮会自动返回到日记列表页面。

3. 日记的添加

为了方便用户使用，也方便测试，可以在 index/home 页面中添加两个按钮式的链接，使其关联到 index/add 页面和 index/manager 页面。

在 index/home 页面增加一个用于存放两个按钮的<view>，修改后的完整代码如下：

```
<!--pages/index/home.wxml-->
<view>
  <navigator url="add" hover-class="navigator-hover">
    <button class="btn" style='float:left' type="primary">新增文章</button>
  </navigator>
  <navigator url="manager" hover-class="navigator-hover">
    <button class="btn" type="default">管理文章</button>
  </navigator>
</view>
<view class="main">
  <view class="list" wx:for="{{items}}" wx:key="id">
    <navigator url="diary?id={{item.id}}" hover-class="navigator-hover">
      <text class="article_title">{{item.title}}</text>
      <text class="article_username">作者：{{item.username}}</text>
      <view>
        <text class="article_date">日期：{{item.date}}</text>
      </view>
    </navigator>
  </view>
</view>
```

在 wxss 页面中新增按钮的样式：

```
.btn{
    width: 50%;
```

```
    font-size:15px;
}
```

修改后的显示效果如图 5-29 所示。

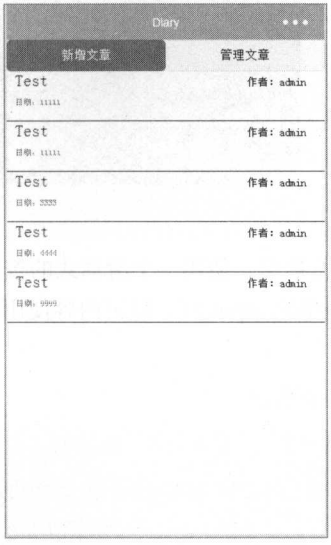


图 5-29 修改后的显示效果

单击“新增文章”按钮，会自动跳转至新增页面，其路由为 index/add。根据系统设计，用户可以在其页面上填写相关的文章内容。也就是说，需要一个可用于提交内容的表单。所以基本的 wxml 页面代码如下所示。

```
<!--pages/index/add.wxml-->
<view class="main">
  <form bindsubmit="formSubmit">
    <view >
      <input name="title" class="article_title" placeholder="日记标题" />
    </view>
    <view class="article_label">
      <label>是否公开：</label>
      <switch name="status" />
    </view>
    <view class="article_date">
      <picker name="date" mode="date" value="{{date}}" start="2015-09-01"
end="2017-09-01" bindchange="bindDateChange">
        <view class="picker">
          选择日期{{date}}
```

```
          </view>
        </picker>
      </view>
      <view class="article_text">
        <textarea   name="text"   bindblur="bindTextAreaBlur"   auto-height
placeholder="日记内容" />
      </view>
      <button style="margin-top:20px" size="mini" formType="submit">增加新日
志</button>
    </form>
  </view>
```

这里使用了一个 form 表单进行提交，使用一个滑动式的按钮设置是否公开，使用一个 time 选择器进行时间的选择。标题使用简单的 input 框，日记内容使用 textare 控件，并设置为自动换长度的方式。

对样式进行调整，具体代码如下所示。

```
/* pages/index/add.wxss */
.main {
  background: #9999ff;
  width: 100%;
  height: 598px;
}
.article_title{
    background-color: #ffffff;
}
.article_label{
    margin-top: 20px;
    background-color: #ffffff;
}
.article_date{
    margin-top: 20px;
    background-color: #ffffff;
}
.article_text{
    margin-top: 20px;
    background-color: #ffffff;
}
```

显示效果如图 5-30 所示。

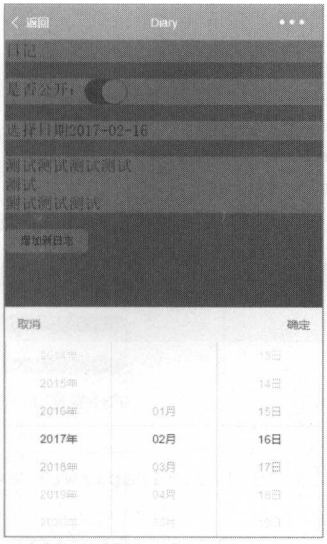


图 5-30 显示效果

对于后台逻辑部分，测试服务器的地址为 http://114.215.29.122:44/api/write。需要将表单的内容发送至服务器。请求参数和说明如表 5-14 所示。

表 5-14 请求参数和说明

参 数	类 型	说 明
token	String	开发者验证 Token
usertoken	String	作者 Token，服务器将会从此 Token 确定其作者
title	String	文章标题
date	String	文章发布日期
text	String	文章内容
status	Int	文章是否为公开，其值为 1 时公开，为 0 则不公开

将请求参数格式化后发送给服务器，服务器会检查是否成功而返回相关的值。完整的 JS 代码如下：

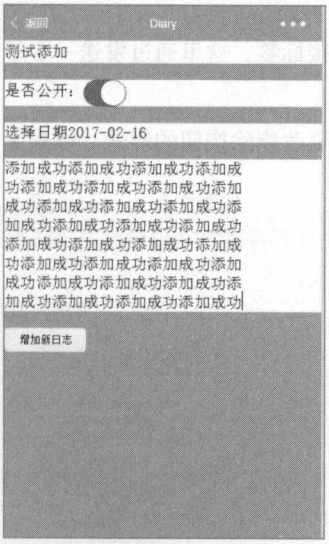
```
//pages/index/add.js
Page({
  data:{},
  onLoad:function(options){
    //页面初始化 options 为页面跳转所带来的参数
  },
   bindDateChange: function(e) {
```

```
    this.setData({
      date: e.detail.value
    })
  },
  formSubmit: function(e) {
    //console.log(e.detail.value)
    var temp;
    if(e.detail.value.status){
       temp=1;
    }else{
      temp=0;
    }
    //页面初始化后发出新的请求
    wx.request({
      url: 'http://114.215.29.122:44/api/write',//仅为示例，并非真实的接口地址
      data: {
        'token': getApp().globalData.userInfo.dev_token,
        'usertoken': wx.getStorageSync('usertoken'),
        'title':e.detail.value.title,
        'date':e.detail.value.date,
        'text':e.detail.value.text,
        'status':temp
      },
      method: 'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        console.log(res.data)
        if (res.data.status == 1) {
          // console.log(JSON.parse(res.data.data))
          wx.redirectTo({
            url: 'home',
          })
        } else {
          wx.showToast({
            title: res.data.message,
            icon: 'loading',
            duration: 2000
          })
        }
      }
```

```
      })
    },
  })
```

注意：对于新增日记的返回值，仅仅返回一个通用返回值。但和其他成功操作 message 为 null 不一致，其 message 中存有添加成功的信息。

添加成功后的显示效果如图 5-31、图 5-32 所示。


返回
Diary
测试添加
是否公开：
选择日期2017-02-16
添加成功添加成功添加成功

```
        <button class="btn" bindtap="" size="mini" type="warn">删除文章
</button>
      </navigator>
      <view>
        <text class="article_date">日期: {{item.date}}</text>
      </view>
    </view>
  </view>
```

增加了一个按钮，并且给这个按钮设置了一个链接标签，这里通过发送一个带参数的请求给该页面，实现不同的功能。

对样式进行简单调整，依旧仿照主页面，只需要更改删除按钮的样式。样式文件的完整代码如下所示。

```
/* pages/index/manager.wxss */
.list{
    height: 60px;
    width: 100%;
    border-bottom: 1px solid #000;
}
.article_title{
    padding: 10px;
    font-size: 20px;
}
.article_date{
    padding: 10px;
    font-size: 10px;
    color: gray
}
.btn{
    width: 120px;
    font-size:15px;
    float:right;
}
```

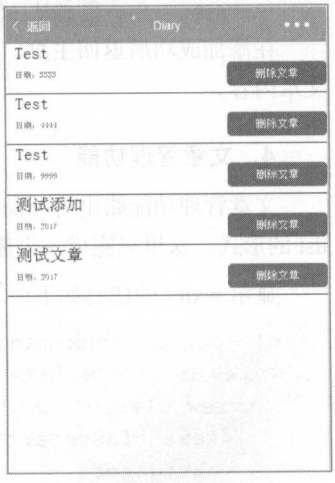


图 5-33　管理页面

页面显示效果如图 5-33 所示。

对于 JS 的处理，这里需要对两个接口进行请求：第一个是在页面载入时，需要请求服务器的获得个人所有日记的接口地址；第二个是当用户单击“删除”按钮时，会访问删除日记接口。

对于载入页面时发出的第一个请求，获得个人所有日记的测试服务器的接口地址为 http://114.215.29.122:44/api/list，需要两个参

数，开发者 Token 和用户的 usertoken，服务器会根据用户的 usertoken 进行值的返回。

删除单篇日记的接口地址为 http://114.215.29.122:44/api/delete，请求包含三个相关的参数，除去开发者 Token 和用户的 usertoken 以外，还需要一个 articleid 进行删除文章的识别。此参数通过链接的形式进行发送请求。通过 onload()获取删除文章的 id，检测此参数是显示该文章列表，还是删除该文章。

```
//pages/index/manager.js
var temp_data;
Page({
  data: {
    items: []
  },
  onLoad: function (e) {
    //页面初始化后发出新的请求
    //检测是否有参数传递，有参数的话执行删除操作，并返回
    if (e.id == null) {
      //刷新页面
      this.getArticle(e);
    } else {
      //删除
      this.delArticle(e)
      setTimeout(this.getArticle(),2000)
    }

  },
  onReady: function () {
    //页面渲染完成
    this.setData({
      items: temp_data
    })
  },
  getArticle: function (e) {
    wx.request({
      url: 'http://114.215.29.122:44/api/list', //仅为示例，并非真实的接口地址
      data: {
        'token': getApp().globalData.userInfo.dev_token,
        'usertoken': wx.getStorageSync('usertoken')
      },
      method: 'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
```

```
    },
    success: function (res) {
      console.log(res.data)
      if (res.data.status == 1) {
        //console.log(JSON.parse(res.data.data))
        temp_data = JSON.parse(res.data.data)
      } else {
        wx.showToast({
          title: res.data.message,
          icon: 'loading',
          duration: 2000
        })
      }
    }
  })
},
delArticle: function (e) {
  console.log(e)
  //页面初始化后发出新的请求
  wx.request({
    url: 'http://114.215.29.122:44/api/delete',//仅为示例，并非真实的接口地址
    data: {
      'token': getApp().globalData.userInfo.dev_token,
      'usertoken': wx.getStorageSync('usertoken'),
      'articleid': e.id
    },
    method: 'POST',
    header: {
      'content-type': 'application/x-www-form-urlencoded'
    },
    success: function (res) {
      console.log(res.data)
      if (res.data.status == 1) {
        // console.log(JSON.parse(res.data.data))
        wx.showToast({
          title: res.data.message,
          icon: 'loading',
          duration: 2000
        })
      } else {
        wx.showToast({
          title: res.data.message,
```

```
            icon: 'loading',
            duration: 2000
          })
        }
      }
    })
  }
})
```

将两次请求都封装成函数，在删除操作后重新获取文章列表，实现页面的自刷新。

注意：和添加文章相同，删除后也会返回 status 为 1，message 中依旧存在提示，删除成功，使其显示两秒。

删除成功，如图 5-34 所示。

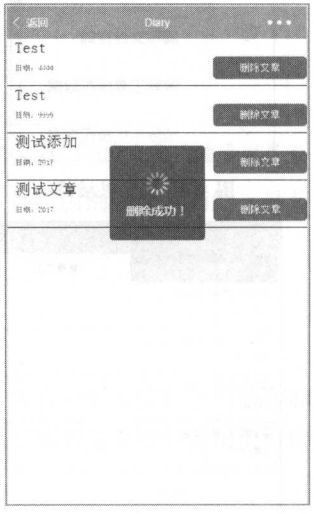


图 5-34 删除成功

到此为止，一个相对完整的交互式的微信小程序开发完成了。当然，对于一个完成的、可以上线的小程序而言，本章的内容还是远远不够的，这只是一个日记 Demo，读者可以尝试在此基础上进行更深层次的开发和功能的迭代。

5.3 小结

本章通过一个比较完整的微信小程序的开发功能，一步步展现了一个小程序的交互步骤和开发流程。从流程图和原型系统设计，到具体的编码和测试，终于完成了一个略微完整的小程序的开发工作。对于这个小程序，也可以用实际操作流程进行简单的测试。

（1）单击“缓存”按钮，删除所有的缓存，如图 5-35 所示。

（2）单击“缓存”按钮上方的“重启”按钮，重启小程序。

（3）进入开屏页，略微等待后出现自动登录加载界面，之后跳转至登录页面。

（4）此时单击“注册”按钮。输入如图 5-36 的测试注册账号和密码，并且重复相同的密码验证。

（5）单击“注册”按钮，提示注册成功，单击左上角的“返回”按钮，返回登录界面，输入此次注册的用户名和密码，如图 5-37 所示。

（6）单击“登录”按钮，成功登录，进入主页面，如图 5-38 所示。

（7）单击“新增文章”按钮，进行一篇新文章的增加，如图 5-39 所示。

（8）单击“增加新日志”按钮，成功增加，重新进入主页，此时，主页出现此篇文章，如图 5-40 所示。

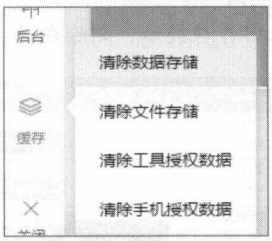


图 5-35　清理缓存

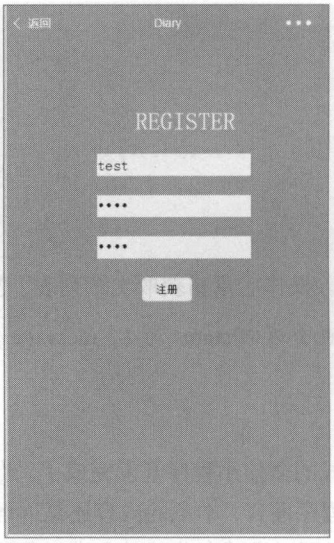


图 5-36　注册账号

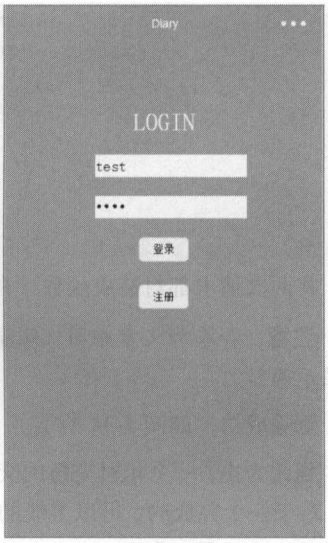


图 5-37　登录账户

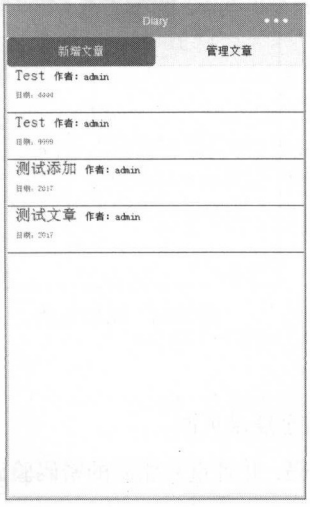


图 5-38　登录后的主页面

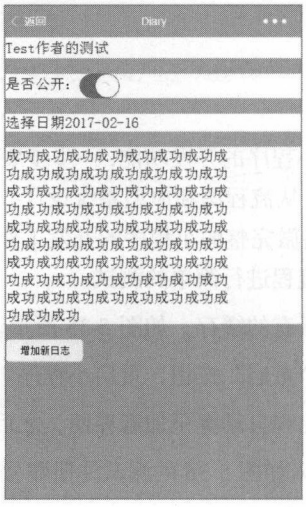


图 5-39　新增文章

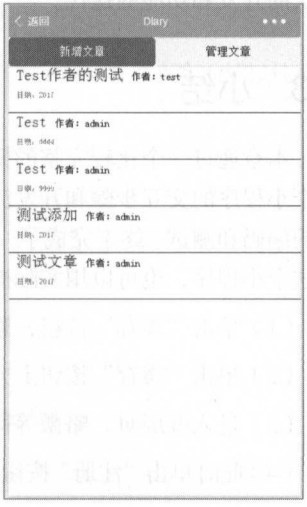


图 5-40　新增文章后的主页

（9）单击此篇文章的链接，进入文章详情页，如图 5-41 所示。

（10）单击“返回”按钮，返回主页，单击“管理文章”按钮，进入 Test 用户的文章管理界面，如图 5-42 所示。

（11）单击“删除文章”按钮，删除此篇文章，此时的管理界面为空，如图 5-43 所示。


返回
Diary
Test作者的测试
发布日期：2017
作者：test
成功成功成功

6 微信提供的 API 解析

小程序框架提供丰富的微信原生 API，可以方便地调起微信提供的能力，如获取用户信息、本地存储、支付功能等。

wx.on 开头的 API 是监听某个事件发生的 API 接口，它接受一个 CALLBACK 函数作为参数。当该事件触发时，会调用 CALLBACK 函数。如未特殊约定，其他 API 接口都接受一个 OBJECT 作为参数。OBJECT 中可以指定 success、fail、complete 来接收接口调用结果，具体的说明如表 6-1 所示。

表 6-1 返回参数说明

参 数 名	类 型	必 填	说 明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

6.1 网络

每个微信小程序需要事先设置一个通信域名，小程序可以跟指定的域名进行网络通信，包括普通 HTTPS 请求（wx.request）、WebSocket 通信（wx.connectSocket）、上传文件（wx.uploadFile）和下载文件（wx.downloadFile）。

注意：网络请求的 referer 是不可以设置的，格式固定为 https://servicewechat.com/{appid}/{version}/page-frame.html，其中{appid}为小程序的 appid，{version}为小程序的版本号，版本号为 0 时表示开发版。

6.1.1 发起请求

wx.request(OBJECT)，其中 wx.request 发起的是 HTTPS 请求，OBJECT 参数说明，根据不同的

需求可以选择添加不同的请求参数，如表 6-2 所示。

表 6-2 参数说明

参 数 名	类 型	必 填	说 明
url	String	是	开发者服务器接口地址
data	Object、String	否	请求的参数
header	Object	否	设置请求的 header，header 中不能设置 Referer
method	String	否	默认为 GET，有效值：OPTIONS、GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT
dataType	String	否	默认为 json，如果设置了 dataType 为 json，则会尝试对响应的数据做一次 JSON.parse
success	Function	否	收到开发者服务成功返回的回调函数，res={data:'开发者服务器返回的内容'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

data 数据是请求接口时上传的数据内容，最终发送给服务器的数据是 String 类型，如果传入的 data 不是 String 类型，会被转换成 String 类型，即对数据进行符合规则的 encode 操作。转换规则如下：

- 对于 header['content-type']为'application/json'的数据，会对数据进行 json 序列化。
- 对于 header['content-type']为'application/x-www-form-urlencoded'的数据，会将数据转换成 querystring（encodeURIComponent(k)=encodeURIComponent(v)&encodeURI Component(k)=encodeURIComponent(v)...）。

注意：对于一般的服务器接口而言，一定要知道其接受的数据类型，否则可能造成请求无效的情况。

【示例 6-1】

```
wx.request({
  url: 'test.php', //仅为示例，并非真实的接口地址
  data: {
     x: '1' ,
     y: '2'
  },
  header: {
      'content-type': 'application/json'
  },
  success: function(res) {
    console.log(res.data)
  }
})
```

上述代码的意义在于，将数据{x:'1',y:'2'}转为 json 发送到 url 地址为 test.php 的地址处，如果请求成功，会在 success 接口处返回相应的值，并在控制台打印出结果。

此操作对于联网的微信小程序而言是非常基础的 API，在上一章的小程序项目中，几乎每一页应用均使用了此 API 携带相关的参数发出请求给服务器端，接收服务器端的返回并实现相关的功能。

6.1.2 上传与下载

1. wx.uploadFile(OBJECT)

将本地资源上传到开发者服务器。如页面通过 wx.chooseImage 等接口获取到一个本地资源的临时文件路径后，可通过此接口将本地资源上传到指定服务器。客户端发起一个 HTTPS POST 请求，其中 content-type 为 multipart/form-data。OBJECT 参数说明如表 6-3 所示。

表 6-3 OBJECT参数说明

参 数	类 型	必 填	说 明
url	String	是	开发者服务器 url
filePath	String	是	要上传文件资源的路径
name	String	是	文件对应的 key，开发者在服务器端通过这个 key 可以获取到文件二进制内容
header	Object	否	HTTP 请求 header，header 中不能设置 Referer
formData	Object	否	HTTP 请求中其他额外的 form data
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-4 所示。

表 6-4 返回参数说明

参 数	类 型	说 明
data	String	开发者服务器返回的数据
statusCode	Number	HTTP 状态码

【示例 6-2】

```
wx.chooseImage({
  success: function(res) {
    var tempFilePaths = res.tempFilePaths
    wx.uploadFile({
      url: 'test.php', //仅为示例，非真实的接口地址
      filePath: tempFilePaths[0],
```

```
      name: 'file',
      formData:{
        'user': 'test'
      },
      success: function(res){
        var data = res.data
        //do something
      }
    })
  }
})
```

对于此页面上传本身而言，其实其本质相当于模拟一个 Web 形式的表单上传方式，使用 HTTP 的 POST 方式进行文件上传。

注意：最大并发限制是 10 个，且默认超时时间和最大超时时间都是 60 秒。

2. wx.downloadFile(OBJECT)

下载文件资源到本地。客户端直接发起一个 HTTP GET 请求，返回文件的本地临时路径。OBJECT 参数说明如表 6-5 所示。

表 6-5 OBJECT参数说明

参　数	类　型	必填	说　明
url	String	是	下载资源的 url
header	Object	否	HTTP 请求 header
success	Function	否	下载成功后以 tempFilePath 的形式传给页面，res = {tempFilePath: '文件的临时路径'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

注意：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需主动调用 wx.saveFile，在小程序下次启动时才能访问得到。

【示例 6-3】

```
wx.downloadFile({
  url: 'test.php', //仅为示例，并非真实的资源
  success: function(res) {
    wx.playVoice({
      filePath: res.tempFilePath
    })
  }
```

```
})
```

注意：最大并发限制是 10 个，且默认超时时间和最大超时时间都是 60 秒。

6.1.3 WebSocket

WebSocket protocol是HTML 5一种新的协议，它实现了浏览器与服务器全双工通信(full-duple)，一开始的握手需要借助 HTTP 请求完成。

1．wx.connectSocket(OBJECT)

创建一个 WebSocket 连接；一个微信小程序同时只能有一个 WebSocket 连接，如果当前已存在一个 WebSocket 连接，会自动关闭该连接，并重新创建一个 WebSocket 连接。

OBJECT 参数说明如表 6-6 所示。

表 6-6　OBJECT参数说明

参　数	类　型	必　填	说　明
url	String	是	开发者服务器接口地址必须是 wss 协议，且域名必须是后台配置的合法域名
data	Object	否	请求的数据
header	Object	否	HTTP header，header 中不能设置 Referer
method	String	否	默认是 GET，有效值为：OPTIONS、GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-4】

```
wx.connectSocket({
  url: 'test.php',
  data:{
    x: '',
    y: ''
  },
  header:{
    'content-type': 'application/json'
  },
  method:"GET"
})
wx.onSocketOpen(CALLBACK)
```

【示例 6-5】

监听 WebSocket 连接打开事件：

```
wx.connectSocket({
  url: 'test.php'
})
wx.onSocketOpen(function(res) {
  console.log('WebSocket 连接已打开！')
})
wx.onSocketError(CALLBACK)
```

【示例 6-6】

监听 WebSocket 错误：

```
wx.connectSocket({
  url: 'test.php'
})
wx.onSocketOpen(function(res){
  console.log('WebSocket 连接已打开！')
})
wx.onSocketError(function(res){
  console.log('WebSocket 连接打开失败，请检查！')
})
wx.sendSocketMessage(OBJECT)
```

2. wx.onSocketOpen

通过 WebSocket 连接发送数据，需要先进行 wx.connectSocket，并在 wx.onSocketOpen 上回调之后才能发送。wx.onSocketOpen 的 OBJECT 参数说明如表 6-7 所示。

表 6-7 OBJECT参数说明

参　数	类　型	必　填	说　明
data	String/ArrayBuffer	是	需要发送的内容
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-7】

监听 WebSocket 接收到服务器的消息事件：

```
var socketOpen = false
```

```
var socketMsgQueue = []
wx.connectSocket({
  url: 'test.php'
})

wx.onSocketOpen(function(res) {
  socketOpen = true
  for (var i = 0; i < socketMsgQueue.length; i++){
    sendSocketMessage(socketMsgQueue[i])
  }
  socketMsgQueue = []
})

function sendSocketMessage(msg) {
  if (socketOpen) {
    wx.sendSocketMessage({
      data:msg
    })
  } else {
    socketMsgQueue.push(msg)
  }
}
wx.onSocketMessage(CALLBACK)
```

CALLBACK 的返回参数说明如表 6-8 所示。

表 6-8　CALLBACK的返回参数说明

参　数	类　型	说　明
data	String/ArrayBuffer	服务器返回的消息

【示例 6-8】

wx.onSocketOpen 的 CALLBACK：

```
wx.connectSocket({
  url: 'test.php'
})

wx.onSocketMessage(function(res) {
  console.log('收到服务器内容：' + res.data)
})
wx.closeSocket()
//关闭 WebSocket 连接
```

```
wx.onSocketClose(CALLBACK)
//监听 WebSocket 关闭

wx.connectSocket({
  url: 'test.php'
})

//注意这里有时序问题
//如果 wx.connectSocket 还没回调 wx.onSocketOpen，而先调用 wx.closeSocket，那么就达不到关闭 WebSocket 的目的
//必须在 WebSocket 打开期间调用 wx.closeSocket，才能关闭
wx.onSocketOpen(function() {
  wx.closeSocket()
})

wx.onSocketClose(function(res) {
  console.log('WebSocket 已关闭！')
})
```

注意：createSocket 连接默认和最大超时时间都是 60 秒，且一个小程序在运行的过程中只能保证有一个 WebSocket 链接。

6.2 媒体

小程序可以调用由微信实现的从本地相册选择图片，或使用相机拍照的 API，还可以操作网络上的图片、音频、文件，甚至对视频进行相关的播放和暂停关闭等功能。

6.2.1 图片

1. wx.chooseImage(OBJECT)

从本地相册选择图片或使用相机拍照，OBJECT 参数说明如表 6-9 所示。

表 6-9 参数说明

参 数	类 型	必 填	说 明
count	Number	否	最多可以选择的图片张数，默认 9
sizeType	StringArray	否	original 原图，compressed 压缩图，默认二者都有
sourceType	StringArray	否	album 从相册选图，camera 使用相机，默认二者都有

续表

参　数	类　型	必　填	说　明
success	Function	是	成功则返回图片的本地文件路径列表 tempFilePaths
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

注意：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需主动调用 wx.saveFile，在小程序下次启动时才能访问到。

【示例 6-9】

```
wx.chooseImage({
  count: 1,                          //默认 9
  sizeType: ['original', 'compressed'],//可以指定是原图还是压缩图，默认二者都有
  sourceType: ['album', 'camera'], //可以指定来源是相册还是相机，默认二者都有
  success: function (res) {
    //返回选定照片的本地文件路径列表，tempFilePath 可以作为 img 标签的 src 属性显示图片
    var tempFilePaths = res.tempFilePaths
  }
})
```

2．wx.previewImage(OBJECT)

预览图片，OBJECT 参数说明如表 6-10 所示。

表 6-10　OBJECT参数说明

参　数	类　型	必　填	说　明
current	String	否	当前显示图片的链接，不填则默认为 urls 的第一张
urls	StringArray	是	需要预览的图片链接列表
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-10】

```
wx.previewImage({
  current: '',  //当前显示图片的 http 链接
  urls: []  //需要预览的图片的 http 链接列表
})
```

3．wx.getImageInfo(OBJECT)

获取图片信息，OBJECT 参数说明如表 6-11 所示。

表 6-11 OBJECT参数说明

参 数	类 型	必 填	说 明
src	String	是	图片的路径，可以是相对路径、临时文件路径、存储文件路径、网络图片路径
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-12 所示。

表 6-12 SUCCESS返回参数说明

参 数	类 型	说 明
width	Number	图片宽度，单位 px
height	Number	图片高度 单位 px
path	String	返回图片的本地路径

【示例 6-11】

```
wx.getImageInfo({
  src: 'images/a.jpg',
  success: function (res) {
    console.log(res.width)
    console.log(res.height)
  }
})

wx.chooseImage({
  success: function (res) {
    wx.getImageInfo({
      src: res.tempFilePaths[0],
      success: function (res) {
        console.log(res.width)
        console.log(res.height)
      }
    })
  }
})
```

6.2.2 录音

1. wx.startRecord(OBJECT)

开始录音。当主动调用 wx.stopRecord，或者录音超过 1 分钟时自动结束录音，返回录音文件的

临时文件路径。当用户离开小程序时，此接口无法调用。OBJECT 参数说明如表 6-13 所示。

注意：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需主动调用 wx.saveFile，在小程序下次启动时才能访问得到。

表 6-13　OBJECT参数说明

参　数	类　型	必　填	说　明
success	Function	否	录音成功后调用，返回录音文件的临时文件路径，res = {tempFilePath: '录音文件的临时路径'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2. wx.stopRecord()

主动调用停止录音。

【示例 6-12】

```
wx.startRecord({
  success: function(res) {
    var tempFilePath = res.tempFilePath
  },
  fail: function(res) {
     //录音失败
  }
})
setTimeout(function() {
  //结束录音
  wx.stopRecord()
}, 10000)
```

注意：wx.startRecord 接口需要用户授权，需兼容用户拒绝授权的场景。

6.2.3　语音控制

1. wx.playVoice(OBJECT)

开始播放语音，同时只允许一个语音文件正在播放，如果前一个语音文件还没播放完，将中断前一个语音播放。OBJECT 参数说明如表 6-14 所示。

表 6-14　OBJECT参数说明

参　数	类　型	必　填	说　明
filePath	String	是	需要播放的语音文件的文件路径

续表

参　数	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-13】

```
wx.startRecord({
  success: function(res) {
    var tempFilePath = res.tempFilePath
    wx.playVoice({
      filePath: tempFilePath,
      complete: function(){
      }
    })
  }
})
```

2. wx.pauseVoice()

暂停正在播放的语音。当再次调用 wx.playVoice 播放同一个文件时，会从暂停处开始播放。如果想从头开始播放，需要先调用 wx.stopVoice。

【示例 6-14】

```
wx.startRecord({
  success: function(res) {
    var tempFilePath = res.tempFilePath
      wx.playVoice({
      filePath: tempFilePath
    })

    setTimeout(function() {
        //暂停播放
      wx.pauseVoice()
    }, 5000)
  }
})
```

3. wx.stopVoice()

结束播放语音。

【示例 6-15】

```
wx.startRecord({
  success: function(res) {
    var tempFilePath = res.tempFilePath
    wx.playVoice({
      filePath:tempFilePath
    })

    setTimeout(function(){
      wx.stopVoice()
    }, 5000)
  }
})
```

6.2.4 音乐控制

1. wx.getBackgroundAudioPlayerState(OBJECT)

获取后台音乐播放状态。OBJECT 参数说明如表 6-15 所示。

表 6-15 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

返回参数说明如表 6-16 所示。

表 6-16 返回参数说明

参 数	说 明
duration	选定音频的长度（单位：s），只有在当前有音乐播放时返回
currentPosition	选定音频的播放位置（单位：s），只有在当前有音乐播放时返回
status	播放状态（2：没有音乐在播放；1：播放中；0：暂停中）
downloadPercent	音频的下载进度（整数，80 代表 80%），只有在当前有音乐播放时返回
dataUrl	歌曲数据链接，只有在当前有音乐播放时返回

【示例 6-16】

```
wx.getBackgroundAudioPlayerState({
   success: function(res) {
      var status = res.status
```

```
        var dataUrl = res.dataUrl
        var currentPosition = res.currentPosition
        var duration = res.duration
        var downloadPercent = res.downloadPercent
    }
})
```

2. wx.playBackgroundAudio(OBJECT)

使用后台播放器播放音乐，对于微信客户端来说，只能同时有一个后台音乐在播放。当用户离开小程序后，音乐将暂停播放；当用户单击“显示在聊天顶部”时，音乐不会暂停播放；当用户在其他小程序占用了音乐播放器，原有小程序内的音乐将停止播放。OBJECT 参数说明如表 6-17 所示。

表 6-17 OBJECT参数说明

参 数	类 型	必 填	说 明
dataUrl	String	是	音乐链接
title	String	否	音乐标题
coverImgUrl	String	否	封面 URL
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-17】

```
wx.playBackgroundAudio({
    dataUrl: '',
    title: '',
    coverImgUrl: ''
})
```

3. wx.pauseBackgroundAudio()

暂停播放音乐。比如：

```
wx.pauseBackgroundAudio()
```

4. wx.seekBackgroundAudio(OBJECT)

控制音乐播放进度。OBJECT 参数说明如表 6-18 所示。

【示例 6-18】

```
wx.seekBackgroundAudio({
    position: 30
})
```

表 6-18　OBJECT参数说明

参　数	类　型	必　填	说　明
position	Number	是	音乐位置，单位秒
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

5．wx.stopBackgroundAudio()

停止播放音乐。比如：

```
wx.stopBackgroundAudio()
```

6．wx.onBackgroundAudioPlay(CALLBACK)

监听音乐播放。

7．wx.onBackgroundAudioPause(CALLBACK)

监听音乐暂停。

8．wx.onBackgroundAudioStop(CALLBACK)

监听音乐停止。

注意：iOS 6.3.30 执行 wx.seekBackgroundAudio 会有短暂延迟。

9．wx.createAudioContext(audioId)

创建并返回 audio 上下文 audioContext 对象。

10．audioContext

audioContext 通过 audioId 跟一个<audio/>组件绑定，通过它可以操作对应的<audio/>组件。audioContext 对象的方法说明如表 6-19 所示。

表 6-19　方法说明

方　法	参　数	说　明
setSrc	src	音频的地址
play	无	播放
pause	无	暂停
seek	position	跳转到指定位置，单位秒

【示例 6-19】

文件操作，对于前端页面的显示，其 wxml 代码为：

```
<!-- audio.wxml -->
<audio  src="{{src}}" id="myAudio" ></audio>

<button type="primary" bindtap="audioPlay">播放</button>
<button type="primary" bindtap="audioPause">暂停</button>
<button type="primary" bindtap="audio14">设置当前播放时间为 14 秒</button>
<button type="primary" bindtap="audioStart">回到开头</button>
```

对于其 JavaScript 逻辑代码为：

```
//audio.js
Page({
  onReady: function (e) {
    //使用 wx.createAudioContext 获取 audio 上下文 context
this.audioCtx=wx.createAudioContext('myAudio')
    this.audioCtx.setSrc('http://ws.stream.qqmusic.qq.com/
M500001VfvsJ21xFqb.mp3?guid=ffffffff82def4af4b12b3cd9337d5e7&uin=346897220&
vkey=6292F51E1E384E06DCBDC9AB7C49FD713D632D313AC4858BACB8DDD29067D3C601481D
36E62053BF8DFEAF74C0A5CCFADD6471160CAF3E6A&fromtag=46')
    this.audioCtx.play()
  },
  data: {
    src: ''
  },
  audioPlay: function () {
    this.audioCtx.play()
  },
  audioPause: function () {
    this.audioCtx.pause()
  },
  audio14: function () {
    this.audioCtx.seek(14)
  },
  audioStart: function () {
    this.audioCtx.seek(0)
  }
})
```

6.2.5 视频控制

1. wx.chooseVideo(OBJECT)

拍摄视频或从手机相册中选视频，返回视频的临时文件路径。OBJECT 参数说明如表 6-20 所示。

表 6-20 OBJECT参数说明

参 数	类 型	必 填	说 明
sourceType	StringArray	否	album 从相册选视频，camera 使用相机拍摄，默认为['album', 'camera']
maxDuration	Number	否	拍摄视频最长拍摄时间，单位秒。最长支持 60 秒
camera	String	否	默认调起的为前置还是后置摄像头。front 为前置，back 为后置，默认 back
success	Function	否	接口调用成功，返回视频文件的临时文件路径，详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

返回参数说明如表 6-21 所示。

表 6-21 返回参数说明

参 数	说 明
tempFilePath	选定视频的临时文件路径
duration	选定视频的时间长度
size	选定视频的数据量大小
height	返回选定视频的长
width	返回选定视频的宽

【示例 6-20】

获取视频。其前台页面的完整代码为：

```
<view class="container">
    <video src="{{src}}"></video>
    <button bindtap="bindButtonTap">获取视频</button>
</view>
```

其 JavaScript 逻辑实现代码为：

```
Page({
    bindButtonTap: function() {
        var that = this
        wx.chooseVideo({
            sourceType: ['album','camera'],
            maxDuration: 60,
```

```
        camera: 'back',
            success: function(res) {
                that.setData({
                    src: res.tempFilePath
                })
            }
        })
    }
})
```

2. wx.createVideoContext(videoId)

创建并返回 video 上下文 videoContext 对象。

3. videoContext

videoContext 通过 videoId 跟一个 video 组件绑定，通过它可以操作一个 video 组件。videoContext 对象的方法列表：

表 6-22 方法说明

方 法	参 数	说 明
play	无	播放
pause	无	暂停
seek	position	跳转到指定位置，单位秒
sendDanmu	danmu	发送弹幕，danmu 包含两个属性 text、color

【示例 6-21】

视频控制。其前台页面的完整代码为：

```
<view class="section tc">
  <video      id="myVideo"      src="http://wxsnsdy.tc.qq.com/105/20210/
snsdyvideodownload?filekey=30280201010421301f0201690402534804102ca905ce620b
1241b726bc41dcff44e00204012882540400&bizid=1023&hy=SH&fileparam=302c0201010
42530230204136ffd93020457e3c4ff02024ef202031e8d7f02030f42400204045a320a0201
000400"  enable-danmu danmu-btn controls></video>
  <view class="btn-area">
    <input bindblur="bindInputBlur"/>
    <button bindtap="bindSendDanmu">发送弹幕</button>
  </view>
</view>
```

其 JavaScript 逻辑实现代码为：

```
function getRandomColor () {
  let rgb = []
  for (let i = 0 ; i < 3; ++i){
    let color = Math.floor(Math.random() * 256).toString(16)
    color = color.length == 1 ? '0' + color : color
    rgb.push(color)
  }
  return '#' + rgb.join('')
}

Page({
  onReady: function (res) {
    this.videoContext = wx.createVideoContext('myVideo')
  },
  inputValue: '',
  bindInputBlur: function(e) {
    this.inputValue = e.detail.value
  },
  bindSendDanmu: function () {
    this.videoContext.sendDanmu({
      text: this.inputValue,
      color: getRandomColor()
    })
  }
})
```

6.2.6 上手实践：获取本地视频和播放控制

这里可以在示例 6-21 的基础上进行更改，结合视频控制的内容，对视频播放和控制进行一次联系和总结。其 wxml 文件的代码如下。

```
<!--index.wxml-->
<view class="container">
  <video src="{{src}}" id="myVideo"></video>
  <view class="btn-area">
    <button bindtap="bindPlay">开始</button>
    <button bindtap="bindStop">停止</button>
    <button bindtap="bindJump">跳转 5s</button>
    <input bindblur="bindInputBlur" />
    <button bindtap="bindSendDanmu">发送弹幕</button>
  </view>
```

```
  <button bindtap="bindButtonTap">获取视频</button>
</view>
```

其逻辑控制页面 JavaScript 的代码如下。

```
//index.js
//获取应用实例
var app = getApp()
function getRandomColor() {
  let rgb = []
  for (let i = 0; i < 3; ++i) {
    let color = Math.floor(Math.random() * 256).toString(16)
    color = color.length == 1 ? '0' + color : color
    rgb.push(color)
  }
  return '#' + rgb.join('')
}
Page({
  onReady: function (res) {
    this.videoContext = wx.createVideoContext('myVideo')
  },
  inputValue: '',
  bindInputBlur: function (e) {
    this.inputValue = e.detail.value
  },
  bindSendDanmu: function () {
    this.videoContext.sendDanmu({
      text: this.inputValue,
      color: getRandomColor()
    })
  },
  bindPlay:function(){
    this.videoContext.play()
  },
  bindStop:function(){
    this.videoContext.pause()
  },
  bindJump:function(){
    this.videoContext.seek(0)
  },
  bindButtonTap: function () {
    var that = this
    wx.chooseVideo({
```

```
      sourceType: ['album', 'camera'],
      maxDuration: 60,
      camera: 'back',
      success: function (res) {
        that.setData({
          src: res.tempFilePath
        })
      }
    })
  }
})
```

视频控制的显示效果如图 6-1 所示。单击获取视频，可以在本地进行一个视频的播放，单击“开始”按钮可以播放此视频。



图 6-1　视频控制的显示效果

6.3　文件

1. wx.saveFile(OBJECT)

保存文件到本地。OBJECT 参数说明如表 6-23 所示。

表 6-23　OBJECT参数说明

参　数	类　型	必　填	说　明
tempFilePath	String	是	需要保存的文件的临时路径
success	Function	否	返回文件的保存路径，res = {savedFilePath: '文件的保存路径'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-22】

```
wx.saveFile({
  success: function(res) {
    var tempFilePath = res.tempFilePath
    wx.saveFile({
      tempFilePath: tempFilePath,
      success: function(res) {
        var savedFilePath = res.savedFilePath
      }
    })
  }
})
```

注意：本地文件存储的大小限制为 10M。

2. wx.getSavedFileList(OBJECT)

获取本地已保存的文件列表，OBJECT 参数说明如表 6-24 所示。

表 6-24 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	否	接口调用成功的回调函数，返回结果见 success 返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

返回参数说明如表 6-25 所示。

表 6-25 返回参数说明

参 数	类 型	说 明
errMsg	String	接口调用结果
fileList	Object Array	文件列表

fileList 中的项目说明如表 6-26 所示。

表 6-26 项目说明

键	类 型	说 明
filePath	String	文件的本地路径
createTime	Number	文件保存时的时间戳，从 1970/01/01 08:00:00 到当前时间的秒数
size	Number	文件大小，单位 B

【示例 6-23】

获取文件列表：

```
wx.getSavedFileList({
  success: function(res) {
    console.log(res.fileList)
  }
})
wx.getSavedFileInfo(OBJECT)
```

3. wx.getSavedFileInfo(OBJECT)

获取本地文件的文件信息，OBJECT 参数说明如表 6-27 所示。

表 6-27　OBJECT参数说明

参数	类　型	必　填	说　明
filePath	String	是	文件路径
success	Function	否	接口调用成功的回调函数，返回结果见 success 返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

filePath 返回参数说明如表 6-28 所示。

表 6-28　返回参数说明

参　数	类　型	说　明
errMsg	String	接口调用结果
size	Number	文件大小，单位 B
createTime	Number	文件的保存时的时间戳，从 1970/01/01 08:00:00 到当前时间的秒数

【示例 6-24】

获取文件信息：

```
wx.getSavedFileInfo({
  filePath: 'wxfile://somefile', //仅作为示例用，非真正的文件路径
  success: function(res) {
    console.log(res.size)
    console.log(res.createTime)
  }
})
```

4．wx.removeSavedFile(OBJECT)

删除本地存储的文件，OBJECT 参数说明如表 6-29 所示。

表 6-29　OBJECT参数说明

参　数	类　型	必　填	说　明
filePath	String	是	需要删除的文件路径
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-25】

删除文件：

```
wx.getSavedFileList({
  success: function(res) {
    if (res.fileList.length > 0){
      wx.removeSavedFile({
        filePath: res.fileList[0].filePath,
        complete: function(res) {
          console.log(res)
        }
      })
    }
  }
})
```

5. wx.openDocument(OBJECT)

新开页面打开文档，支持格式：doc、xls、ppt、pdf、docx、xlsx、pptx，OBJECT 参数说明如表 6-30 所示。

表 6-30 OBJECT参数说明

参 数	说 明	必填	说 明
filePath	String	是	文件路径，可通过 downFile 获得
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-26】

新建打开文件的页面：

```
wx.downloadFile({
  url: 'http://example.com/somefile.pdf',
  success: function (res) {
    var filePath = res.tempFilePath
    wx.openDocument({
      filePath: filePath,
      success: function (res) {
        console.log('打开文档成功')
      }
    })
  }
})
```

6.4 数据缓存

每个微信小程序都可以有自己的本地缓存，可以通过 wx.setStorage（wx.setStorageSync）、wx.getStorage（wx.getStorageSync）、wx.clearStorage（wx.clearStorageSync）对本地缓存进行设置、获取和清理。本地缓存最大为 10MB。

注意：localStorage 是永久存储的，但是不建议将关键信息全部存在 localStorage 上，以防用户换设备的情况。

1. wx.setStorage(OBJECT)

将数据存储在本地缓存中指定的 key 中，会覆盖掉原来该 key 对应的内容，这是一个异步接口。OBJECT 参数说明如表 6-31 所示。

表 6-31 OBJECT参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中的指定的 key
data	Object/String	是	需要存储的内容
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-27】

缓存到指定 key：

```
wx.setStorage({
  key:"key",
  data:"value"
})
```

2. wx.setStorageSync(KEY,DATA)

将 data 存储在本地缓存指定的 key 中会覆盖掉原来该 key 对应的内容，这是一个同步接口。参数说明如表 6-32 所示。

表 6-32 参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中指定的 key
data	Object/String	是	需要存储的内容

【示例 6-28】

同步缓存到指定 key：

```
try {
    wx.setStorageSync('key', 'value')
} catch (e) {
}
```

3. wx.getStorage(OBJECT)

从本地缓存中异步获取指定 key 对应的内容。OBJECT 参数说明如表 6-33 所示。

表 6-33 OBJECT参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中指定的 key
success	Function	是	接口调用的回调函数，res = {data: key 对应的内容}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-29】

异步获取指定 key：

```
wx.getStorage({
  key: 'key',
  success: function(res) {
     console.log(res.data)
  }
})
```

4. wx.getStorageSync(KEY)

从本地缓存中同步获取指定 key 对应的内容。

【示例 6-30】

获取指定 key：

```
try {
  var value = wx.getStorageSync('key')
  if (value) {
     //返回值
  }
} catch (e) {
  //捕捉错误时执行的内容
}
```

5. wx.getStorageInfo(OBJECT)

异步获取当前 storage 的相关信息。OBJECT 参数说明如表 6-34 所示。

表 6-34 OBJECT参数说明

参　数	类　型	必　填	说　明
success	Function	是	接口调用的回调函数，详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

返回参数说明如表 6-35 所示。

表 6-35 返回参数说明

参　数	类　型	说　明
keys	String Array	当前 storage 中所有的 key
currentSize	Number	当前占用的空间大小，单位 KB
limitSize	Number	限制的空间大小，单位 KB

【示例 6-31】

异步获取：

```
wx.getStorageInfo({
  success: function(res) {
    console.log(res.keys)
    console.log(res.currentSize)
    console.log(res.limitSize)
  }
})
```

6. wx.getStorageInfoSync

同步获取当前 storage 的相关信息。

【示例 6-32】

```
try {
  var res = wx.getStorageInfoSync()
  console.log(res.keys)
  console.log(res.currentSize)
  console.log(res.limitSize)
} catch (e) {
  //捕捉错误时执行的内容
}
```

7. wx.removeStorage(OBJECT)

从本地缓存中异步移除指定 key。OBJECT 参数说明如表 6-36 所示。

表 6-36 OBJECT参数说明

参 数	类 型	必 填	说 明
key	String	是	本地缓存中指定的 key
success	Function	是	接口调用的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-33】

移除指定 key：

```
wx.removeStorage({
  key: 'key',
  success: function(res) {
    console.log(res.data)
  }
})
```

8. wx.removeStorageSync(key)

从本地缓存中同步移除指定 key，参数说明如表 6-37 所示。

表 6-37 参数说明

参 数	类 型	必 填	说 明
key	String	是	本地缓存中指定的 key

【示例 6-34】

移除指定 key：

```
try {
  wx.removeStorageSync('key')
} catch (e) {
  //捕捉错误时执行的内容
}
```

9. wx.clearStorage()

清理本地数据缓存，比如：

```
wx.clearStorage()
```

10. wx.clearStorageSync()

同步清理本地数据缓存。

【示例 6-35】

```
try {
    wx.clearStorageSync()
} catch(e) {
  //捕捉错误时执行的内容
}
```

注意：对于其本地数据存储的大小限制仍为 10MB。

6.5 位置

6.5.1 获取位置

1. wx.getLocation(OBJECT)

获取当前的地理位置、速度。当用户离开小程序后，此接口无法调用；当用户单击“显示在聊天顶部”时，此接口可继续调用。

OBJECT 参数说明如表 6-38 所示。

表 6-38 OBJECT参数说明

参　数	类　型	必　填	说　明
type	String	否	默认为 wgs84，返回 GPS 坐标，gcj02 返回可用于 wx.openLocation 的坐标
success	Function	是	接口调用成功的回调函数，返回内容详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-39 所示。

表 6-39 SUCCESS返回参数说明

参　数	说　明
latitude	纬度，浮点数，范围为-90~90，负数表示南纬
longitude	经度，浮点数，范围为-180~180，负数表示西经
speed	速度，浮点数，单位 m/s
accuracy	位置的精确度

【示例 6-36】

```
wx.getLocation({
  type: 'wgs84',
  success: function(res) {
    var latitude = res.latitude
    var longitude = res.longitude
    var speed = res.speed
    var accuracy = res.accuracy
  }
})
```

2. wx.chooseLocation(OBJECT)

打开地图选择位置，OBJECT 参数说明如表 6-40 所示。

表 6-40 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	是	接口调用成功的回调函数，返回内容详见返回参数说明
cancel	Function	否	用户取消时调用
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-41 所示。

表 6-41 SUCCESS返回参数说明

参 数	说 明
name	位置名称
address	详细地址
latitude	纬度，浮点数，范围为-90~90，负数表示南纬
longitude	经度，浮点数，范围为-180~180，负数表示西经

6.5.2 查看位置

wx.openLocation(OBJECT)，使用微信内置地图查看位置，OBEJCT 参数说明如表 6-42 所示。

表 6-42 OBJECT参数说明

参 数	类 型	必 填	说 明
latitude	Float	是	纬度，范围为-90~90，负数表示南纬
longitude	Float	是	经度，范围为-180~180，负数表示西经
scale	INT	否	缩放比例，范围 5~18，默认为 18

续表

参　数	类　型	必　填	说　明
name	String	否	位置名
address	String	否	地址的详细说明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-37】

```
wx.getLocation({
  type: 'gcj02',      //返回可以用于 wx.openLocation 的经纬度
  success: function(res) {
    var latitude = res.latitude
    var longitude = res.longitude
    wx.openLocation({
      latitude: latitude,
      longitude: longitude,
      scale: 28
    })
  }
})
```

注意：iOS6.3.30 中 type 参数不生效，只会返回 wgs84 类型的坐标信息且 wx.getLocation、wx.chooseLocation 接口需要用户授权，请兼容用户拒绝授权的场景。

6.5.3　地图组件控制

1．wx.createMapContext(mapId)

创建并返回 map 上下文 mapContext 对象。

2．mapContext

mapContext 通过 mapId 跟一个<map/>组件绑定，通过它可以操作对应的<map/>组件。mapContext 对象的方法说明如表 6-43 所示。

表 6-43　mapcontext对象方法说明

方　法	参　数	说　明
getCenterLocation	OBJECT	获取当前地图中心的经纬度，返回的是 gcj02 坐标系，可以用于 wx.openLocation
moveToLocation	无	将地图中心移动到当前定位点，需要配合 map 组件的 show-location 使用

OBJECT 参数说明如表 6-44 所示。

表 6-44 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	否	接口调用成功的回调函数，res={longitude:"经度",latitude:"纬度"}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-38】

其前台显示完整代码：

```
<!-- map.wxml -->
<map id="myMap" show-location />

<button type="primary" bindtap="getCenterLocation">获取位置</button>
<button type="primary" bindtap="moveToLocation">移动位置</button>
```

其逻辑处理代码：

```
//map.js
Page({
  onReady: function (e) {
    //使用 wx.createMapContext 获取 map 上下文
    this.mapCtx = wx.createMapContext('myMap')
  },
  getCenterLocation: function () {
    this.mapCtx.getCenterLocation({
      success: function(res){
        console.log(res.longitude)
        console.log(res.latitude)
      }
    })
  },
  moveToLocation: function () {
    this.mapCtx.moveToLocation()
  }
})
```

6.5.4 上手实践：地图控制和查看

通过对于地图插件和 JavaScript 的编写，获得用户当前的位置信息，并且可以让用户进行查看大地图。

其 wxml 代码如下：

```
<!--index.wxml-->
<view class="container">
  <map id="myMap" show-location />
  <button bindtap="getCenterLocation">获取位置</button>
  <button bindtap="moveToLocation">移动位置</button>
  <button bindtap="moveToLocation">查看大地图</button>
</view>
```

其 JavaScript 代码如下：

```
//index.js
//获取应用实例
var app = getApp()
Page({
  onReady: function (e) {
    //使用 wx.createMapContext 获取 map 上下文
    this.mapCtx = wx.createMapContext('myMap')
  },
  getCenterLocation: function () {
    this.mapCtx.getCenterLocation({
      success: function (res) {
        console.log(res.longitude)
        console.log(res.latitude)
      }
    })
  },
  moveToLocation: function () {
    this.mapCtx.moveToLocation()
  },
  showBigMap:function(){
    wx.getLocation({
      type: 'gcj02', //返回可以用于 wx.openLocation 的经纬度
      success: function (res) {
        var latitude = res.latitude
        var longitude = res.longitude
        wx.openLocation({
          latitude: latitude,
          longitude: longitude,
          scale: 28
        })
      }
```

```
        })
    }
})
```

地图控制的显示效果如图 6-2 所示。单击“查看大地图”可以跳转至地图页面，如图 6-3 所示。

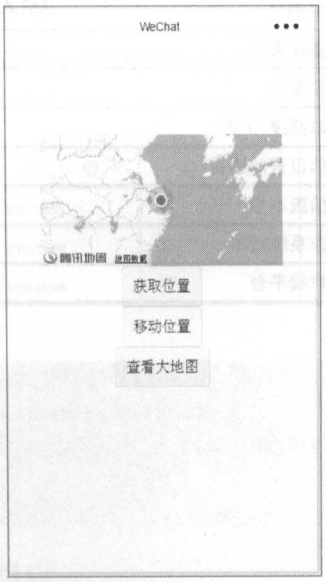


图 6-2 地图控制的显示效果



图 6-3 查看大地图

6.6 设备

微信端的权限可以通过提供的相关 API 实现获取系统信息、设备信息等。

6.6.1 系统信息

1. wx.getSystemInfo(OBJECT)

获取系统信息，OBJECT 参数说明如表 6-45 所示。

表 6-45 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	是	接口调用成功的回调
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 回调参数说明如表 6-46 所示。

表 6-46　SUCCESS回调参数说明

属　性	说　明
model	手机型号
pixelRatio	设备像素比
windowWidth	窗口宽度
windowHeight	窗口高度
language	微信设置的语言
version	微信版本号
system	操作系统版本
platform	客户端平台

【示例 6-39】

```
wx.getSystemInfo({
  success: function(res) {
    console.log(res.model)
    console.log(res.pixelRatio)
    console.log(res.windowWidth)
    console.log(res.windowHeight)
    console.log(res.language)
    console.log(res.version)
    console.log(res.platform)
  }
})
```

2. wx.getSystemInfoSync()

获取系统信息同步接口。

【示例 6-40】

```
try {
  var res = wx.getSystemInfoSync()
  console.log(res.model)
  console.log(res.pixelRatio)
  console.log(res.windowWidth)
  console.log(res.windowHeight)
  console.log(res.language)
  console.log(res.version)
  console.log(res.platform)
```

```
} catch (e) {
  //捕获错误时执行的内容
}
```

6.6.2 网络状态

wx.getNetworkType(OBJECT)获取网络类型，OBJECT 参数说明如表 6-47 所示。

表 6-47 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	是	接口调用成功，返回网络类型 networkType
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-41】

```
wx.getNetworkType({
  success: function(res) {
    var networkType = res.networkType //返回网络类型 2g、3g、4g、Wi-Fi
  }
})
```

6.6.3 重力感应

wx.onAccelerometerChange(CALLBACK)监听重力感应数据，频率为 5 次/秒。CALLBACK 返回参数如表 6-48 所示。

表 6-48 CALLBACK返回参数说明

参 数	类 型	说 明
x	Number	X 轴
y	Number	Y 轴
z	Number	Z 轴

【示例 6-42】

```
wx.onAccelerometerChange(function(res) {
  console.log(res.x)
  console.log(res.y)
  console.log(res.z)
})
```

6.6.4 罗盘

wx.onCompassChange(CALLBACK)监听罗盘数据，频率 5 次/秒。CALLBACK 返回参数如表 6-49 所示。

表 6-49 CALLBACK返回参数说明

参 数	类 型	说 明
direction	Number	面对的方向度数

【示例 6-43】

```
wx.onCompassChange(function (res) {
  console.log(res.direction)
})
```

6.6.5 拨打电话

wx.makePhoneCall(OBJECT)，OBJECT 参数说明如表 6-50 所示。

表 6-50 OBJECT参数说明

参 数	类 型	必 填	说 明
phoneNumber	String	是	需要拨打的电话号码
success	Function	否	接口调用成功的回调
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-44】

```
wx.makePhoneCall({
  phoneNumber: '1340000' //仅为示例，并非真实的电话号码
})
```

6.6.6 扫码

wx.scanCode(OBJECT)调起客户端扫码界面，扫码成功后返回对应的结果。OBJECT 参数说明如表 6-51 所示。

表 6-51 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	否	接口调用成功的回调函数，返回内容详见返回参数说明。
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-52 所示。

表 6-52 SUCCESS返回参数说明

参 数	说 明
result	所扫码的内容
scanType	所扫码的类型
charSet	所扫码的字符集
path	当所扫的码为当前小程序的合法二维码时，会返回此字段，内容为二维码携带的 path

【示例 6-45】

```
wx.scanCode({
  success: (res) => {
    console.log(res)
  }
})
```

6.6.7 上手实践：获取所有手机信息和网络状态

通过上手实践，可以总结对于手机信息和网络信息的获取，巩固对于微信小程序各种状态以及数据绑定的学习和练习。

其前台代码如下：

```
<!--index.wxml-->
<view class="container">
  <text>language:{{phonedata.language}}</text>
  <text>model:{{phonedata.model}}</text>
  <text>pixelRatio:{{phonedata.pixelRatio}}</text>
  <text>platform: {{phonedata.platform}}</text>
  <text>system: {{phonedata.system}}</text>
  <text>version: {{phonedata.version}}</text>
  <text>windowHeight: {{phonedata.windowHeight}}</text>
  <text>windowWidth: {{phonedata.windowWidth}}</text>
  <text>line:{{line}}</text>
</view>
```

其 JavaScript 的逻辑代码如下：

```
//index.js
//获取应用实例
var app = getApp()
var networkType
Page({
```

```
  data: {
    phonedata: {},
    line: {}
  },
  //事件处理函数
  bindViewTap: function () {
    wx.navigateTo({
      url: '../logs/logs'
    })
  },
  onLoad: function () {
    wx.getNetworkType({
      success: function (res) {
        //返回网络类型，有效值
        //Wi-Fi/2g/3g/4g/unknown(Android下不常见的网络类型)/none(无网络)
        networkType = res.networkType
      }
    })
    this.setData({
      phonedata: wx.getSystemInfoSync(),
      line: networkType
    }
    )
    console.log(this.data.phonedata)
  },
  onReady:function(){
      this.setData({
    line: networkType
    }
    )
  }
})
```

设备与网络信息如图 6-4 所示。

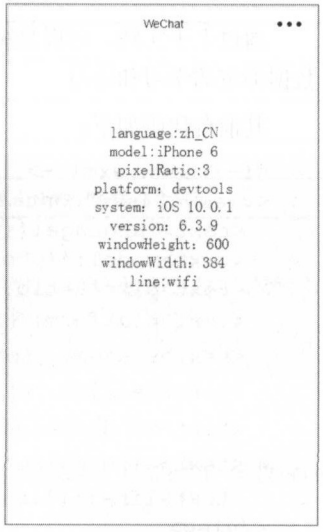


图 6-4　设备与网络信息

6.7　界面

6.7.1　交互反馈

1. wx.showToast(OBJECT)

wx.showToast(OBJECT)为显示消息提示框，OBJECT 参数说明如表 6-53 所示。

表 6-53 OBJECT参数说明

参 数	类 型	必 填	说 明
title	String	是	提示的内容
icon	String	否	图标，只支持"success""loading"
duration	Number	否	提示的延迟时间，单位为毫秒，默认为 1500，最大为 10000
mask	Boolean	否	是否显示透明蒙层，防止触摸穿透，默认 false
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-46】

```
wx.showToast({
  title: '成功',
  icon: 'success',
  duration: 2000
})
```

2. wx.hideToast()

wx.hideToast()为隐藏消息提示框。

【示例 6-47】

```
wx.showToast({
  title: '加载中',
  icon: 'loading',
  duration: 10000
})

setTimeout(function(){
  wx.hideToast()
},2000)
```

3. wx.showModal(OBJECT)

wx.showModal(OBJECT)为显示模态弹窗，OBJECT 参数说明如表 6-54 所示。

表 6-54 OBJECT参数说明

参 数	类 型	必 填	说 明
title	String	是	提示的标题
content	String	是	提示的内容
showCancel	Boolean	否	是否显示取消按钮，默认为 true

续表

参　数	类　型	必　填	说　明
cancelText	String	否	取消按钮的文字，默认为"取消"，最多 4 个字符
cancelColor	HexColor	否	取消按钮的文字颜色，默认为"#000000"
confirmText	String	否	确定按钮的文字，默认为"确定"，最多 4 个字符
confirmColor	HexColor	否	确定按钮的文字颜色，默认为"#3CC51F"
success	Function	否	接口调用成功的回调函数，当返回 res.confirm 为 true 时，表示用户单击“确定”按钮
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-48】

```
wx.showModal({
  title: '提示',
  content: '这是一个模态弹窗',
  success: function(res) {
    if (res.confirm) {
      console.log('用户单击确定')
    }
  }
})
```

4．wx.showActionSheet(OBJECT)

wx.showActionSheet(OBJECT)为显示操作菜单，OBJECT 参数说明如表 6-55 所示。

表 6-55　OBJECT参数说明

参　数	类　型	必　填	说　明
itemList	String Array	是	按钮的文字数组，数组长度最大为 6 个
itemColor	HexColor	否	按钮的文字颜色，默认为"#000000"
success	Function	否	接口调用成功的回调函数，详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-56 所示。

表 6-56　SUCCESS返回参数说明

参　数	类　型	说　明
tapIndex	Number	用户单击的按钮，按从上到下的顺序，从 0 开始

【示例 6-49】

```
wx.showActionSheet({
  itemList: ['A', 'B', 'C'],
  success: function(res) {
    console.log(res.tapIndex)
  },
  fail: function(res) {
    console.log(res.errMsg)
  }
})
```

注意：这里存在一个小 bug，在 Android 6.3.30 中，wx.showModal 返回的 confirm 一直为 true，wx.showActionSheet 单击“取消”或“蒙层”时，回调失败，错误 errMsg 为"showActionSheet:fail cancel"。

6.7.2 设置导航条

wx.setNavigationBarTitle(OBJECT)为动态设置当前页面的标题，OBJECT 参数说明如表 6-57 所示。

表 6-57 OBJECT参数说明

参 数	类 型	必 填	说 明
title	String	是	页面标题
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-50】

```
wx.setNavigationBarTitle({
  title: '当前页面'
})
```

2. wx.showNavigationBarLoading()

wx.showNavigationBarLoading()为在当前页面显示导航条加载动画。

3. wx.hideNavigationBarLoading()

wx.hideNavigationBarLoading()为隐藏导航条加载动画。

6.7.3 导航

1. wx.navigateTo(OBJECT)

wx.navigateTo(OBJECT)为保留当前页面，跳转到应用内的某个页面，使用 wx.navigateBack 可以返回到原页面。

OBJECT 参数说明如表 6-58 所示。

表 6-58 OBJECT参数说明

参　数	类　型	必　填	说　明
url	String	是	需要跳转的应用内非 tabBar 的页面的路径，路径后可以带参数。参数与路径之间使用?分隔，参数键与参数值用=相连，不同参数用&分隔，如'path?key=value&key2=value2'
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-51】

```
wx.navigateTo({
  url: 'test?id=1'
})
//test.js
Page({
  onLoad: function(option){
    console.log(option.query)
  }
})
```

注意：为了不让用户在使用小程序时造成困扰，我们规定页面路径只能是五层，请尽量避免多层级的交互方式。

2. wx.redirectTo(OBJECT)

wx.redirectTo(OBJECT)为关闭当前页面，跳转到应用内的某个页面。OBJECT 参数说明如表 6-59 所示。

表 6-59 OBJECT参数说明

参　数	类　型	必　填	说　明
url	String	是	需要跳转的应用内非 tabBar 的页面的路径，路径后可以带参数。参数与路径之间使用?分隔，参数键与参数值用=相连，不同参数用&分隔，如'path?key=value&key2=value2'

续表

参　数	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-52】

```
wx.redirectTo({
  url: 'test?id=1'
})
```

3. wx.switchTab(OBJECT)

wx.switchTab(OBJECT)为跳转到 tabBar 页面，同时关闭其他所有非 tabBar 页面。OBJECT 参数说明表 6-60 所示。

表 6-60　OBJECT参数说明

参　数	类　型	必　填	说　明
url	String	是	需要跳转的应用内非 tabBar 的页面的路径，路径后可以带参数。参数与路径之间使用?分隔，参数键与参数值用=相连，不同参数用&分隔，如'path?key=value&key2=value2'
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-53】

```
{
  "tabBar": {
    "list": [{
      "pagePath": "index",
      "text": "首页"
    },{
      "pagePath": "other",
      "text": "其他"
    }]
  }
}
wx.switchTab({
  url: '/index'
})
```

4．wx.navigateBack(OBJECT)

关闭当前页面，返回上一页或多级页面。可通过 getCurrentPages() 获取当前的页面栈，决定需要返回几层。OBJECT 参数说明表 6-61 所示。

表 6-61　OBJECT参数说明

参　数	类　型	默 认 值	说　明
delta	Number	1	返回的页面数，如果 delta 大于现有页面数，则返回到首页
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-54】

调用 navigateTo 跳转时，调用该方法的页面会被加入堆栈，而 redirectTo 方法则不会，参见下方示例代码：

```
//此处是 A 页面
wx.navigateTo({
  url: 'B?id=1'
})

//此处是 B 页面
wx.navigateTo({
  url: 'C?id=1'
})

//在 C 页面内 navigateBack，将返回 A 页面
wx.navigateBack({
  delta: 2
})
```

注意：wx.navigateTo 和 wx.redirectTo 不允许跳转到 tabbar 页面，只能用 wx.switchTab 跳转到 tabbar 页面。

6.7.4　动画

wx.createAnimation(OBJECT)。创建一个动画实例 animation。用调用实例的方法来描述动画。最后通过动画实例的 export 方法导出动画数据传递给组件的 animation 属性。OBJECT 参数说明表 6-62 所示。

注意：export 方法每次调用后会清掉之前的动画操作。

表 6-62 OBJECT参数说明

参 数	类 型	必 填	说 明
duration	Integer	否	动画持续时间，单位 ms，默认值 400
timingFunction	String	否	定义动画的效果，默认值"linear"，有效值"linear""ease""ease-in""ease-in-out" "ease-out""step-start""step-end"
delay	Integer	否	动画延迟时间，单位 ms，默认值 0
transformOrigin	String	否	设置 transform-origin，默认为"50%50%0"

【示例 6-55】

动画实例：

```
var animation = wx.createAnimation({
  transformOrigin: "50% 50%",
  duration: 1000,
  timingFunction: "ease",
  delay: 0
})
animation
```

动画实例可以调用以下方法来描述动画，调用结束后会返回自身，支持链式调用的写法。动画的样式如表 6-63 所示。

表 6-63 动画的样式

方 法	参 数	说 明
opacity	value	透明度，参数范围 0~1
backgroundColor	color	颜色值
width	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
height	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
top	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
left	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
bottom	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
right	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值

旋转样式如表 6-64 所示。

表 6-64 旋转样式

方 法	参 数	说 明
rotate	Deg	deg 的范围-180~180，从原点顺时针旋转一个 deg 角度
rotateX	Deg	deg 的范围-180~180，在 X 轴旋转一个 deg 角度

续表

方　法	参　数	说　明
rotateY	Deg	deg 的范围-180~180，在 Y 轴旋转一个 deg 角度
rotateZ	Deg	deg 的范围-180~180，在 Z 轴旋转一个 deg 角度
rotate3d	(x,y,z,deg)	同 transform-function rotate3d

缩放样式如表 6-65 所示。

表 6-65　方法说明

方　法	参　数	说　明
scale	sx,[sy]	一个参数时，表示在 X 轴、Y 轴同时缩放 sx 倍数；两个参数表示在 X 轴缩放 sx 倍数，在 Y 轴缩放 sy 倍数
scaleX	sx	在 X 轴缩放 sx 倍数
scaleY	sy	在 Y 轴缩放 sy 倍数
scaleZ	sz	在 Z 轴缩放 sy 倍数
scale3d	(sx,sy,sz)	在 X 轴缩放 sx 倍数，在 Y 轴缩放 sy 倍数，在 Z 轴缩放 sz 倍数

偏移样式如表 6-66 所示。

表 6-66　偏移样式

方　法	参　数	说　明
translate	tx,[ty]	一个参数时，表示在 X 轴偏移 tx，单位 px；两个参数时，表示在 X 轴偏移 tx，在 Y 轴偏移 ty，单位 px
translateX	tx	在 X 轴偏移 tx，单位 px
translateY	ty	在 Y 轴偏移 tx，单位 px
translateZ	tz	在 Z 轴偏移 tx，单位 px
translate3d	(tx,ty,tz)	在 X 轴偏移 tx，在 Y 轴偏移 ty，在 Z 轴偏移 tz，单位 px

倾斜样式如表 6-67 所示。

表 6-67　倾斜样式

方　法	参　数	说　明
skew	ax,[ay]	参数范围–180~180；一个参数时，Y 轴坐标不变，X 轴坐标沿顺时针倾斜 ax° ；两个参数时，分别在 X 轴倾斜 ax° ，在 Y 轴倾斜 ay°
skewX	ax	参数范围–180~180；Y 轴坐标不变，X 轴坐标沿顺时针倾斜 ax°
skewY	ay	参数范围–180~180；X 轴坐标不变，Y 轴坐标沿顺时针倾斜 ay°

矩阵变形样式如表 6-68 所示。

表 6-68 矩阵变形样式

方 法	参 数	说 明
matrix	(a,b,c,d,tx,ty)	矩阵类型
matrix3d	matrix3d(a1, b1, c1, d1, a2, b2, c2, d2, a3, b3, c3, d3, a4, b4, c4, d4)	多维矩阵类型

调用动画操作方法后要调用 step()来表示一组动画完成，可以在一组动画中调用任意多个动画方法，一组动画中的所有动画会同时开始，一组动画完成后才会进行下一组动画。step 可以传入一个跟 wx.createAnimation()一样的配置参数，用于指定当前组动画的配置。

【示例 6-56】

动画队列，其前台显示界面代码为：

```
<view animation="{{animationData}}" style="background:red;height:100rpx;
width:100rpx"></view>
```

其逻辑页面代码为：

```
Page({
  data: {
    animationData: {}
  },
  onShow: function(){
    var animation = wx.createAnimation({
      duration: 1000,
        timingFunction: 'ease',
    })

    this.animation = animation

    animation.scale(2,2).rotate(45).step()

    this.setData({
      animationData:animation.export()
    })

    setTimeout(function() {
      animation.translate(30).step()
      this.setData({
        animationData:animation.export()
      })
    }.bind(this), 1000)
  },
```

```
  rotateAndScale: function () {
    //旋转同时放大
    this.animation.rotate(45).scale(2, 2).step()
    this.setData({
      animationData: this.animation.export()
    })
  },
  rotateThenScale: function () {
    //先旋转后放大
    this.animation.rotate(45).step()
    this.animation.scale(2, 2).step()
    this.setData({
      animationData: this.animation.export()
    })
  },
  rotateAndScaleThenTranslate: function () {
    //先旋转同时放大，然后平移
    this.animation.rotate(45).scale(2, 2).step()
    this.animation.translate(100, 100).step({ duration: 1000 })
    this.setData({
      animationData: this.animation.export()
    })
  }
})
```

注意：iOS/Android 6.3.30 中通过 step()分隔动画，只有第一步动画能生效。

6.7.5 下拉刷新

1. onPullDownRefresh

在 Page 中定义 onPullDownRefresh 处理函数，监听该页面用户下拉刷新事件，需要在 config 的 window 选项中开启 enablePullDownRefresh。当处理完数据刷新后，wx.stopPullDownRefresh 可以停止当前页面的下拉刷新。

2. wx.stopPullDownRefresh()

wx.stopPullDownRefresh()为停止当前页面下拉刷新。

【示例 6-57】

```
Page({
  onPullDownRefresh: function(){
```

```
      wx.stopPullDownRefresh()
    }
  })
```

6.7.6 上手实践：带有控制的动画

对动画和节点的控制对于一个内容展示应用并不是很重要的功能，但是如果想要开发一个动画页面或小微信游戏，那么它是非常重要的一部分。在这个应用里，将会实现对于一个节点的动画控制功能，单击不同的按钮会展示不同的效果。

其前台代码如下：

```
<!--index.wxml-->
<view    animation="{{animationData}}"    style="background:red;height:
100rpx;width:100rpx"></view>
<button bindtap="TranslateLeft">左</button>
<button bindtap="TranslateRight">右</button>
<button bindtap="rotateAndScale">旋转</button>
```

其 JavaScript 逻辑控制代码如下：

```
//index.js
//获取应用实例
var app = getApp()
Page({
  data: {
    animationData: {}
  },
  onShow: function(){
    var animation = wx.createAnimation({
      duration: 1000,
        timingFunction: 'ease',
    })

    this.animation = animation
    this.setData({
      animationData:animation.export()
    })

    setTimeout(function() {
      animation.translate(30).step()
      this.setData({
        animationData:animation.export()
      })
```

```
    }.bind(this), 1000)
  },
  rotateAndScale: function () {
    //旋转
    this.animation.rotate(45).step()
    this.setData({
      animationData: this.animation.export()
    })
  },
  TranslateLeft: function () {
    //左
    this.animation.translate(100, 0).step({ duration: 1000 })
    this.setData({
      animationData: this.animation.export()
    })
  },
  TranslateRight: function () {
    //平移
    this.animation.translate(300, 0).step({ duration: 1000 })
    this.setData({
      animationData: this.animation.export()
    })
  }
})
```

动画控制的显示效果如图 6-5 所示。

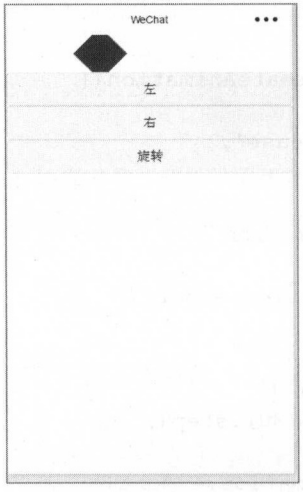


图 6-5　动画控制的显示效果

6.8 绘图

绘图 API 的接口如表 6-69 所示。

表 6-69 绘图API的接口

方 法	说 明
createCanvasContext	创建 canvas 绘图上下文（指定 canvasId）
createContext(不推荐使用)	创建 canvas 绘图上下文
drawCanvas(不推荐使用)	进行绘图
canvasToTempFilePath	导出图片
context	对象的方法列表

绘制颜色、样式、阴影的方法如表 6-70 所示。

表 6-70 绘制颜色、样式、阴影的方法

方 法	说 明
setFillStyle	设置填充样式
setStrokeStyle	设置线条样式
setShadow	设置阴影

绘制渐变的方法如表 6-71 所示。

表 6-71 绘制渐变的方法

方 法	说 明
createLinearGradient	创建一个线性渐变
createCircularGradient	创建一个圆形渐变
addColorStop	在渐变中的某一点添加一个颜色变化

绘制线条样式的方法如表 6-72 所示。

表 6-72 绘制线条样式的方法

方 法	说 明
setLineWidth	设置线条宽度
setLineCap	设置线条端点的样式
setLineJoin	设置两线相交处的样式
setMiterLimit	设置最大倾斜

绘制矩形的方法如表 6-73 所示。

表 6-73　绘制矩形的方法

方　法	说　明
rect	创建一个矩形
fillRect	填充一个矩形
strokeRect	画一个矩形（不填充）
clearRect	在给定的矩形区域内，清除画布上的像素

绘制路径的方法如表 6-74 所示。

表 6-74　绘制路径的方法

方　法	说　明
fill	对当前路径进行填充
stroke	对当前路径进行描边
beginPath	开始一个路径
closePath	关闭一个路径
moveTo	把路径移动到画布中的指定点，但不创建线条
lineTo	添加一个新点，然后在画布中创建从该点到最后指定点的线条
arc	添加一个弧形路径到当前路径，顺时针绘制
quadraticCurveTo	创建二次方贝塞尔曲线
bezierCurveTo	创建三次方贝塞尔曲线

绘制变形的方法如表 6-75 所示。

表 6-75　绘制变形的方法

方　法	说　明
scale	对横坐标和纵坐标进行缩放
rotate	对坐标轴进行顺时针旋转
translate	对坐标原点进行缩放

绘制文字的方法如表 6-76 所示。

表 6-76　绘制文字的方法

方　法	说　明
fillText	在画布上绘制被填充的文本
setFontSize	设置字体大小

绘制图片的方法如表 6-77 所示。

表 6-77 绘制图片的方法

方 法	说 明
drawImage	在画布上绘制图像

混合方法如表 6-78 所示。

表 6-78 混合方法

方 法	说 明
setGlobalAlpha	设置全局画笔透明度

其他方法如表 6-79。

表 6-79 其他方法

方 法	说 明
save	保存当前绘图上下文
restore	恢复之前保存过的绘图上下文
draw	进行绘图
getActions(不推荐使用)	获取当前 context 上存储的绘图动作
clearActions(不推荐使用)	清空当前的存储绘图动作

6.8.1 在 canvas 上画图

所有在<canvas/>中的画图必须用 JavaScript 完成。

【示例 6-58】

简单画图，WXML（我们在接下来的例子中如无特殊声明都会用这个 WXML 为模板，不再重复）：

```
<canvas canvas-id="myCanvas" style="border: 1px solid;"/>
```

JS（我们在接下来的例子中会将 JS 放在 onLoad 中）：

```
const ctx = wx.createCanvasContext('myCanvas')
ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 75)
ctx.draw()
```

1. 创建一个 canvas 绘图上下文

创建一个 canvas 绘图上下文，CanvasContext。CanvasContext 是小程序内建的一个对象，有一些绘图的方法：

```
const ctx = wx.createCanvasContext('myCanvas')
```

2. 使用 canvas 绘图上下文进行绘图描述

我们来描述要在 canvas 中绘制什么内容。设置绘图上下文的填充色为红色：

```
ctx.setFillStyle('red')
```

用 fillRect(x, y, width, height)方法画一个矩形，填充为刚刚设置的红色：

```
ctx.fillRect(10, 10, 150, 75)
```

3. 画图

告诉<canvas/>组件要将刚才的描述绘制上去：

```
ctx.draw()
```

结果是一个红色的矩形框图。

6.8.2 canvas 坐标系

canvas是在一个二维的网格当中，左上角的坐标为(0, 0)。在前面的演示中我们用了方法 fillRect(0, 0, 150, 75)，其含义为：从左上角(0, 0)开始，画一个像素为 150 × 75 的矩形。

【示例 6-59】

可以在<canvas/>中加上一些事件，来观测它的坐标系，此示例前台代码如下：

```
<canvas canvas-id="myCanvas"
  style="margin: 5px; border:1px solid #d3d3d3;"
  bindtouchstart="start"
  bindtouchmove="move"
  bindtouchend="end"/>

<view hidden="{{hidden}}">
  Coordinates: ({{x}}, {{y}})
</view>
```

其逻辑页面代码为：

```
Page({
  data: {
    x: 0,
    y: 0,
    hidden: true
  },
  start: function(e) {
    this.setData({
      hidden: false,
```

```
      x: e.touches[0].x,
      y: e.touches[0].y
    })
  },
  move: function(e) {
    this.setData({
      x: e.touches[0].x,
      y: e.touches[0].y
    })
  },
  end: function(e) {
    this.setData({
      hidden: true
    })
  }
})
```

当鼠标或是模拟触摸放到 canvas 中，就会在下边显示出触碰点的坐标。

6.8.3 渐变

渐变能用于填充一个矩形、圆、线、文字等。填充色可以不固定为某一种颜色，形成渐变的效果。微信提供了两种颜色渐变的方式：

- createLinearGradient(x, y, x1, y1)：创建一个线性的渐变。
- createCircularGradient(x, y, r)：创建一个从圆心开始的渐变。

一旦创建了一个渐变对象，必须添加两个颜色渐变点，即初始化的颜色值和结束时的颜色值：

- addColorStop(position, color)方法：用于指定颜色渐变点的位置和颜色，位置必须位于 0 到 1 之间。
- 可以用 setFillStyle()和 setStrokeStyle()方法设置渐变，然后进行画图描述。

【示例 6-60】

使用 createLinearGradient()：

```
const ctx = wx.createCanvasContext('myCanvas')

//创建一个线性的渐变
const grd = ctx.createLinearGradient(0, 0, 200, 0)
grd.addColorStop(0, 'red')
grd.addColorStop(1, 'white')

//填充渐变
```

```
ctx.setFillStyle(grd)
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

【示例 6-61】

使用 createCircularGradient()：

```
const ctx = wx.createCanvasContext('myCanvas')

//创建一个从圆心开始的渐变
const grd = ctx.createCircularGradient(75, 50, 50)
grd.addColorStop(0, 'red')
grd.addColorStop(1, 'white')

//填充渐变
ctx.setFillStyle(grd)
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

1. createLinearGradient

创建一个线性的渐变颜色需要使用 addColorStop() 来指定渐变点，至少要两个。createLinearGradient 参数如表 6-80 所示。

表 6-80 createLinearGradient参数

参　数	类　型	定　义
x0	Number	起点的 x 坐标
y0	Number	起点的 y 坐标
x1	Number	终点的 x 坐标
y1	Number	终点的 y 坐标

【示例 6-62】

创建一个线性的渐变颜色：

```
const ctx = wx.createCanvasContext('myCanvas')

//创建一个线性的渐变颜色
const grd = ctx.createLinearGradient(0, 0, 200, 0)
grd.addColorStop(0, 'red')
grd.addColorStop(1, 'white')

//填充渐变
ctx.setFillStyle(grd)
```

```
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

2. createCircularGradient

用来创建一个圆形的渐变颜色，起点在圆心，终点在圆环，且需要使用 addColorStop()来指定渐变点，至少要两个。

createCircularGradient 参数如表 6-81 所示。

表 6-81 createCircularGradient参数

参 数	类 型	定 义
x	Number	圆心的 x 坐标
y	Number	圆心的 y 坐标
r	Number	圆的半径

【示例 6-63】

创建一个圆形的渐变颜色：

```
const ctx = wx.createCanvasContext('myCanvas')

//创建一个圆形的渐变颜色
const grd = ctx.createCircularGradient(75, 50, 50)
grd.addColorStop(0, 'red')
grd.addColorStop(1, 'white')

//填充渐变
ctx.setFillStyle(grd)
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

3. addColorStop

创建一个颜色的渐变点。小于最小 stop 的部分会按最小 stop 的 color 来渲染，大于最大 stop 的部分会按最大 stop 的 color 来渲染。如果需要使用 addColorStop()来指定渐变点，至少要两个。

addColorStop 参数如表 6-82 所示。

表 6-82 addColorStop参数

参 数	类 型	定 义
stop	Number(0-1)	表示渐变点在起点和终点的位置
color	Color	渐变点的颜色

【示例 6-64】

颜色的渐变：

```
const ctx = wx.createCanvasContext('myCanvas')

//创建圆形渐变点
const grd = ctx.createLinearGradient(30, 10, 120, 10)
grd.addColorStop(0, 'red')
grd.addColorStop(0.16, 'orange')
grd.addColorStop(0.33, 'yellow')
grd.addColorStop(0.5, 'green')
grd.addColorStop(0.66, 'cyan')
grd.addColorStop(0.83, 'blue')
grd.addColorStop(1, 'purple')

//填充渐变
ctx.setFillStyle(grd)
ctx.fillRect(10, 10, 150, 80)
ctx.draw()
```

6.8.4 上手实践：使用 canvas 绘制笑脸

通过最常见的点可以绘制出直线，通过不同的直线可以绘制出曲线和面，通过基础的图形变化，可以将一个简单的画布变得丰富多彩。本节的小示例便是在 canvas 的基础上绘制出一个笑脸。

其前台代码如下：

```
<!--index.wxml-->
<canvas style="width: 300px; height: 200px;" canvas-id="firstCanvas">
</canvas>
<!-- 当使用绝对定位时，文档流后边的 canvas 的显示层级高于前边的 canvas -->
<canvas style="width: 400px; height: 500px;" canvas-id="secondCanvas">
</canvas>
<!-- 因为 canvas-id 与前一个 canvas 重复，该 canvas 不会显示，并会发送一个错误事件
到 AppService -->
<canvas style="width: 400px; height: 500px;" canvas-id="secondCanvas"
binderror="canvasIdErrorCallback"></canvas>
```

其 JavaScript 的控制绘制代码如下：

```
//index.js
```

```
//获取应用实例
var app = getApp()
Page({
  canvasIdErrorCallback: function (e) {
    console.error(e.detail.errMsg)
  },
  onReady: function (e) {

    //使用 wx.createContext 获取绘图上下文 context
    var context = wx.createContext()

    context.setStrokeStyle("#00ff00")
    context.setLineWidth(5)
    context.rect(0, 0, 200, 200)
    context.stroke()
    context.setStrokeStyle("#ff0000")
    context.setLineWidth(2)
    context.moveTo(160, 100)
    context.arc(100, 100, 60, 0, 2 * Math.PI, true)
    context.moveTo(140, 100)
    context.arc(100, 100, 40, 0, Math.PI, false)
    context.moveTo(85, 80)
    context.arc(80, 80, 5, 0, 2 * Math.PI, true)
    context.moveTo(125, 80)
    context.arc(120, 80, 5, 0, 2 * Math.PI, true)
    context.stroke()

    //调用 wx.drawCanvas，通过 canvasId 指定在哪张画布上绘制，通过 actions 指定绘制行为
    wx.drawCanvas({
      canvasId: 'firstCanvas',
      actions: context.getActions() //获取绘图动作数组
    })
  }
})
```

绘制笑脸如图 6-6 所示。

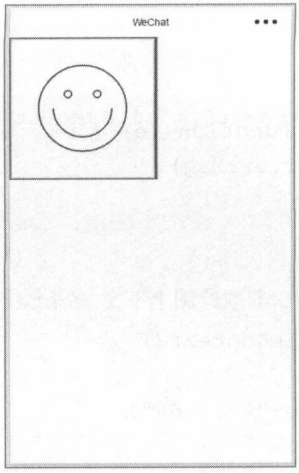


图 6-6　绘制笑脸

6.9　开放接口

6.9.1　登录

1. wx.login(OBJECT)

调用接口获取登录凭证（code）进而换取用户登录态信息，包括用户的唯一标识（openid）及本次登录的会话密钥（session_key）。用户数据的加密解密通信都依赖于会话密钥完成。OBJECT 参数说明如表 6-83 所示。

表 6-83　OBJECT参数说明

参数名	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-84 所示。

表 6-84　success返回参数说明

参数名	类　型	说　明
errMsg	String	调用结果
code	String	用户允许登录后，回调内容会带上 code（有效期五分钟），开发者需要将 code 发送到开发者服务器后台，使用 code 换取 session_keyapi，将 code 换成 openid 和 session_key

【示例 6-65】

```
//app.js
App({
  onLaunch: function() {
    wx.login({
      success: function(res) {
        if (res.code) {
          //发起网络请求
          wx.request({
            url: 'https://test.com/onLogin',
            data: {
              code: res.code
            }
          })
        } else {
          console.log('获取用户登录态失败! ' + res.errMsg)
        }
      }
    });
  }
})
```

2. code 换取 session_key

这是一个 HTTPS 接口，开发者服务器使用登录凭证 code 获取 session_key 和 openid。其中 session_key 是对用户数据进行加密签名的密钥。为了自身应用安全，session_key 不应该在网络上传输。接口地址如下所示：

```
https://api.weixin.qq.com/sns/jscode2session?appid=APPID&secret=SECRET&js_code=JSCODE&grant_type=authorization_code
```

请求参数如表 6-85 所示。

表 6-85 请求参数

参 数	必 填	说 明
appid	是	小程序唯一标识
secret	是	小程序的 appsecret
js_code	是	登录时获取的 code
grant_type	是	填写为 authorization_code

返回参数如表 6-86 所示。

表 6-86　返回参数

参　数	说　明
openid	用户唯一标识
session_key	会话密钥

返回说明：

```
//正常返回的 json 数据包
{
      "openid": "OPENID",
      "session_key": "SESSIONKEY"
}
//错误时返回 json 数据包(示例为 code 无效)
{
    "errcode": 40029,
    "errmsg": "invalid code"
}
```

3. 登录态维护

通过 wx.login()获取到用户登录态之后，需要维护登录态。开发者要注意不应该直接把 session_key、openid 等字段作为用户的标识或者 session 的标识，而应该自己派发一个 session 登录态（请参考登录时序图）。对于开发者自己生成的 session，应该保证其安全性且不应该设置较长的过期时间。session 派发到小程序客户端之后，可将其存储在 storage 上，用于后续通信使用。

4. 登录时序图

wx.checkSession(OBJECT)可以检查登录态是否过期，参数如表 6-87 所示。

表 6-87　发送参数说明

参 数 名	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数，登录态未过期
fail	Function	否	接口调用失败的回调函数，登录态已过期
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 6-66】

登录状态：

```
wx.checkSession({
  success: function(){
    //登录状态未过期
```

```
    },
    fail: function(){
      //登录状态过期
      wx.login()
    }
  })
```

注意：iOS/Android 6.3.30，在 App.onLaunch 调用 wx.login 会出现异常。

5. 用户数据的签名验证

为了确保开放接口返回用户数据的安全性，微信会对明文数据进行签名。开发者可以根据业务需要对数据包进行签名校验，确保数据的完整性。

签名校验算法涉及用户的 session_key，通过 wx.login 登录流程获取用户 session_key，并自行维护与应用自身登录态的对应关系。

通过调用接口（如 wx.getUserInfo）获取数据时，接口会同时返回 rawData、signature，其中 signature=sha1(rawData+session_key)。

开发者将 signature、rawData 发送到开发者服务器进行校验。服务器利用用户对应的 session_key 使用相同的算法计算出签名 signature2，比对 signature 与 signature2 即可校验数据的完整性。

【示例 6-67】

wx.getUserInfo 的数据校验，接口返回 rawData：

```
{
  "nickName": "Band",
  "gender": 1,
  "language": "zh_CN",
  "city": "Guangzhou",
  "province": "Guangdong",
  "country": "CN",
  "avatarUrl":          "http://wx.qlogo.cn/mmopen/vi_32/1vZvI39NWFQ9XM4Lt
QpFrQJ1xlgZxx3w7bQxKARol6503Iuswjjn6nIGBiaycAjAtpujxyzYsrztuuICqIM5ibXQ/0"
}
```

用户的 session-key：

```
HyVFkGl5F5OQWJZZaNzBBg==
```

所以，用于签名的字符串为：

```
{"nickName":"Band","gender":1,"language":"zh_CN","city":"Guangzhou","pro
vince":"Guangdong","country":"CN","avatarUrl":"http://wx.qlogo.cn/mmopen/vi
_32/1vZvI39NWFQ9XM4LtQpFrQJ1xlgZxx3w7bQxKARol6503Iuswjjn6nIGBiaycAjAtpujxyz
```

```
YsrztuuICqIM5ibXQ/0"}HyVFkGl5F5OQWJZZaNzBBg==
```

使用 sha1 得到的结果为：

```
75e81ceda165f4ffa64f4068af58c64b8f54b88c
```

6. 加密数据解密算法

接口如果涉及敏感数据（如 wx.getUserInfo 当中的 openId 和 unionId），接口的明文内容将不包含这些敏感数据。开发者如需获取敏感数据，要对接口返回的加密数据（encryptedData）进行对称解密。解密算法如下：

- 对称解密使用的算法为 AES-128-CBC，数据采用 PKCS#7 填充。
- 对称解密的目标密文为 Base64_Decode(encryptedData)。
- 对称解密密钥 aeskey=Base64_Decode(session_key)，aeskey 是 16 字节。
- 对称解密算法初始向量 ***iv*** 会在数据接口中返回。

另外，为了应用能校验数据的有效性，我们会对敏感数据加上数据水印（watermark），watermark 参数说明如表 6-88 所示。

表 6-88　watermark参数说明

参　数	类　型	说　明
watermark	OBJECT	数据水印
appid	String	敏感数据归属 appid，开发者可校验此参数与自身 appid 是否一致
timestamp	DateInt	敏感数据获取的时间戳，开发者可以用于数据时效性校验

【示例 6-68】

接口 wx.getUserInfo 敏感数据当中的 watermark：

```
{
    "openId": "OPENID",
    "nickName": "NICKNAME",
    "gender": GENDER,
    "city": "CITY",
    "province": "PROVINCE",
    "country": "COUNTRY",
    "avatarUrl": "AVATARURL",
    "unionId": "UNIONID",
    "watermark":
    {
        "appid":"APPID",
        "timestamp":TIMESTAMP
```

```
    }
}
```

注意：此前提供的加密数据（encryptData）以及对应的加密算法将被弃用，请开发者不要再依赖旧逻辑。

6.9.2 用户信息

wx.getUserInfo(OBJECT)可以获取用户信息，需要先调用 wx.login 接口，OBJECT 参数说明如表 6-89 所示。

表 6-89 OBJECT发送参数说明

参数名	类型	必填	说明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 6-90 所示。

表 6-90 success返回参数说明

参数	类型	说明
userInfo	OBJECT	用户信息对象，不包含 openid 等敏感信息
rawData	String	不包括敏感信息的原始数据字符串，用于计算签名
signature	String	使用 sha1(rawData+sessionkey)得到字符串，用于校验用户信息
encryptedData	String	包括敏感数据在内的完整用户信息的加密数据，详细见加密数据的解密算法
iv	String	加密算法的初始向量，详细见加密数据的解密算法

【示例 6-69】

获取用户信息：

```
wx.getUserInfo({
  success: function(res) {
    var userInfo = res.userInfo
    var nickName = userInfo.nickName
    var avatarUrl = userInfo.avatarUrl
    var gender = userInfo.gender //性别 0：未知、1：男、2：女
    var province = userInfo.province
    var city = userInfo.city
    var country = userInfo.country
  }
```

```
})
```

encryptedData 解密后为以下 json 结构：

```
{
    "openId": "OPENID",
    "nickName": "NICKNAME",
    "gender": GENDER,
    "city": "CITY",
    "province": "PROVINCE",
    "country": "COUNTRY",
    "avatarUrl": "AVATARURL",
    "unionId": "UNIONID",
    "watermark":
    {
       "appid":"APPID",
    "timestamp":TIMESTAMP
    }
}
```

UnionID 机制说明：如果开发者拥有多个移动应用、网站应用和公众账号（包括小程序），可通过 unionid 来区分用户的唯一性，因为只要是同一个微信开放平台账号下的移动应用、网站应用和公众账号（包括小程序），用户的 unionid 是唯一的。换句话说，同一用户，对同一个微信开放平台下的不同应用，unionid 是相同的。

注意：wx.getUserInfo 接口需要用户授权，请兼容用户拒绝授权的场景。

6.9.3 微信支付

wx.requestPayment(OBJECT)可以发起微信支付，OBJECT 参数说明如表 6-91 所示。

表 6-91 OBJECT参数说明

参　数	类　型	必填	说　明
timeStamp	String	是	时间戳从 1970 年 1 月 1 日 00:00:00 至今的秒数，即当前的时间
nonceStr	String	是	随机字符串，长度为 32 个字符以下
package	String	是	统一下单接口返回的 prepay_id 参数值，提交格式如 prepay_id=*
signType	String	是	签名算法，暂支持 MD5
paySign	String	是	签名，具体签名方案参见微信公众号支付帮助文档
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

回调结果如表 6-92 所示。

表 6-92 回调结果

回调类型	errMsg	说 明
success	requestPayment:ok	调用支付成功
fail	requestPayment:fail cancel	用户取消支付
fail	requestPayment:fail (detail message)	调用支付失败，其中 detail message 为后台返回的详细失败原因

【示例 6-70】

微信支付：

```
wx.requestPayment({
  'timeStamp': '',
  'nonceStr': '',
  'package': '',
  'signType': 'MD5',
  'paySign': '',
  'success':function(res){
  },
  'fail':function(res){
  }
})
```

注意：在 6.5.2 及之前版本中，用户取消支付不会触发 fail 回调，只会触发 complete 回调，回调 errMsg 为'requestPayment:cancel'。

6.9.4 模板消息

基于微信的通知渠道，微信为开发者提供了可以高效触达用户的模板消息能力，以便实现服务的闭环并提供更佳的体验。

- 模板推送位置：服务通知。
- 模板下发条件：用户本人在微信体系内与页面有交互行为后触发，详见下发条件说明。
- 模板跳转能力：单击查看仅能跳转下发模板的该账号的各个页面。

1. 使用说明

获取模板 id，登录 https://mp.weixin.qq.com 获取模板，如果没有合适的模板，可以申请添加新模板，审核通过后可使用，详见模板审核说明。

页面的<form/>组件，当属性 report-submit 为 true 时，可以声明为需发模板消息，此时单击按钮提交表单可以获取 formId，用于发送模板消息。当用户完成支付行为，可以获取 prepay_id 用于发送模板消息。

2. 调用接口下发模板消息

（1）获取 access_token

access_token 是全局唯一接口调用凭据，开发者调用各接口时都需使用 access_token，请妥善保存。access_token 的存储至少要保留 512 个字符空间。access_token 的有效期目前为两个小时，需定时刷新，重复获取将导致上次获取的 access_token 失效。

公众平台的 API 调用所需的 access_token 的使用及生成方式说明：

- 为了保密 appsecrect，第三方需要一个 access_token 获取和刷新的中控服务器。而其他业务逻辑服务器所使用的 access_token 均来自于该中控服务器，不应该各自去刷新，否则会造成 access_token 覆盖而影响业务。
- access_token 的有效期通过返回的 expires_in 来传达，目前有效期是 7200 秒之内的值。中控服务器需要根据这个有效时间提前去刷新新 access_token。在刷新过程中，中控服务器对外输出的依然是老 access_token，此时公众平台后台会保证在刷新短时间内，新老 access_token 都可用，这保证了第三方业务的平滑过渡。
- access_token 的有效时间可能会在未来有调整，所以中控服务器不仅需要内部定时主动刷新，还需要提供被动刷新 access_token 的接口，这样便于业务服务器在 API 调用获知 access_token 已超时的情况下，可以触发 access_token 的刷新流程。

开发者可以使用 AppID 和 AppSecret 调用本接口来获取 access_token。AppID 和 AppSecret 可从“微信公众平台官网→设置→开发设置”中获得（需要已经绑定成为开发者，且账号没有异常状态）。AppSecret 生成后请自行保存，因为在公众平台每次生成查看都会导致 AppSecret 被重置。注意调用所有微信接口时均需使用 HTTPS 协议。如果第三方不使用中控服务器，而是选择各个业务逻辑点各自去刷新 access_token，那么就可能产生冲突，导致服务不稳定。

接口地址：

```
https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=APPID&secret=APPSECRET
```

HTTP 请求方式：GET。参数说明如表 6-93 所示。

表 6-93　参数说明

参　数	必　填	说　明
grant_type	是	获取 access_token，填写 client_credential
appid	是	第三方用户唯一凭证
secret	是	第三方用户唯一凭证密钥，即 appsecret

返回参数说明如表 6-94 所示。

表 6-94 返回参数说明

参 数	说 明
access_token	获取到的凭证
expires_in	凭证有效时间，单位：秒

正常情况下，微信会返回下述 json 数据包给开发者：

```
{"access_token": "ACCESS_TOKEN", "expires_in": 7200}
```

错误时微信会返回错误码等信息，json 数据包示例如下（该示例为 AppID 无效错误）：

```
{"errcode": 40013, "errmsg": "invalid appid"}
```

（2）发送模板消息

接口地址（ACCESS_TOKEN 需换成上文获取到的 access_token）：

```
https://api.weixin.qq.com/cgi-bin/message/wxopen/template/send?access_token=ACCESS_TOKEN
```

HTTP 请求方式：POST。POST 参数说明如表 6-95 所示。

表 6-95 POST参数说明

参 数	必 填	说 明
touser	是	接收者（用户）的 openid
template_id	是	所需下发的模板消息的 id
page	否	单击模板卡片后的跳转页面，仅限本小程序内的页面。支持带参数，（示例 index?foo=bar）。该字段不填则模板无跳转
form_id	是	表单提交场景下，为 submit 事件带上的 formId；支付场景下，为本次支付的 prepay_id
value	是	模板内容，不填则下发空模板
color	否	模板内容字体的颜色，不填默认黑色
emphasis_keyword	否	模板需要放大的关键词，不填则默认无放大

【示例 6-71】

发送模板消息：

```
{
  "touser": "OPENID",
  "template_id": "TEMPLATE_ID",
  "page": "index",
  "form_id": "FORMID",
  "data": {
      "keyword1": {
          "value": "339208499",
```

```
            "color": "#173177"
        },
        "keyword2": {
            "value": "2015年01月05日 12:30",
            "color": "#173177"
        },
        "keyword3": {
            "value": "粤海喜来登酒店",
            "color": "#173177"
        } ,
        "keyword4": {
            "value": "广州市天河区天河路208号",
            "color": "#173177"
        }
    },
    "emphasis_keyword": "keyword1.DATA"
}
```

返回码说明：在调用模板消息接口后，会返回 json 数据包。

正常时的返回 json 数据包示例：

```
{
  "errcode": 0,
  "errmsg": "ok",
}
```

错误时会返回错误码信息，如表 6-96 所示。

表 6-96　错误码信息

返 回 码	说　明
40037	template_id 不正确
41028	form_id 不正确，或者过期
41029	form_id 已被使用
41030	page 不正确
45009	接口调用超过限额（目前默认每个账号日调用限额为 100 万）

3. 下发条件说明

- 支付：当用户在小程序内完成过支付行为，可允许开发者向用户在 7 天内推送有限条数的模板消息（1 次支付可下发 1 条，多次支付下发条数独立，互相不影响）。
- 提交表单：当用户在小程序内发生过提交表单行为且该表单声明为要发模板消息的，开发者

需要向用户提供服务时，可允许开发者向用户在 7 天内推送有限条数的模板消息（1 次提交表单可下发 1 条，多次提交下发条数独立，相互不影响）。

注意：微信 6.5.2 及以上版本支持模板功能，低于该版本将无法收到模板消息。

6.9.5 客服消息

1. 接收消息和事件

在页面中使用<contact-button/>可以显示进入客服会话按钮。

当用户在客服会话发送消息（或进行某些特定的用户操作引发的事件推送时），微信服务器会将消息（或事件）的数据包（json 或者 XML 格式）POST 请求开发者填写 URL。开发者收到请求后可以使用发送客服消息接口进行异步回复。

微信服务器在将用户的消息发给小程序的开发者服务器地址（开发设置处配置）后，微信服务器在 5 秒内收不到响应会断掉连接，并且重新发起请求，总共重试三次，如果在调试中，发现用户无法收到响应的消息，可以检查是否是消息处理超时。关于重试的消息排重，有 msgid 的消息推荐使用 msgid 排重。事件类型消息推荐使用 FromUserName+ CreateTime 排重。

服务器收到请求必须做出下述回复，这样微信服务器才不会对此作任何处理，并且不会发起重试，否则，将出现严重的错误注意：

- 直接回复 success（推荐方式）。
- 直接回复空串（指字节长度为 0 的空字符串，而不是结构体中 content 字段的内容为空）。

一旦遇到以下情况，微信会在小程序会话中向用户下发系统提示“该小程序客服暂时无法提供服务，请稍后再试”：

- 开发者在 5 秒内未回复任何内容。
- 开发者回复了异常数据。

如果开发者希望增强安全性，可以在开发者中心开启消息加密，这样用户发给小程序的消息以及小程序回复用户的消息都会继续加密。

各消息类型的推送 json、XML 数据包结构如下：

- 文本消息。

用户在客服会话中发送文本消息时将产生两种格式的数据包。

XML 格式的数据包：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
```

```
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1482048670</CreateTime>
    <MsgType><![CDATA[text]]></MsgType>
    <Content><![CDATA[this is a test]]></Content>
    <MsgId>1234567890123456</MsgId>
</xml>
```

json 格式的数据包：

```
{
    "ToUserName": "toUser",
    "FromUserName": "fromUser",
    "CreateTime": 1482048670,
    "MsgType": "text",
    "Content": "this is a test",
    "MsgId": 1234567890123456
}
```

参数说明如表 6-97 所示。

表 6-97　参数说明

参　数	说　明
ToUserName	小程序的原始 ID
FromUserName	发送者的 openid
CreateTime	消息创建时间（整型）
MsgType	text
Content	文本消息内容
MsgId	消息 id，64 位整型

- 图片消息。

用户在客服会话中发送图片消息时将产生两种格式的数据包。

XML 格式的数据包：

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1482048670</CreateTime>
    <MsgType><![CDATA[image]]></MsgType>
    <PicUrl><![CDATA[this is a url]]></PicUrl>
    <MediaId><![CDATA[media_id]]></MediaId>
    <MsgId>1234567890123456</MsgId>
</xml>
```

json 格式的数据包：

```
{
    "ToUserName": "toUser",
    "FromUserName": "fromUser",
    "CreateTime": 1482048670,
    "MsgType": "image",
    "PicUrl": "this is a url",
    "MediaId": "media_id",
    "MsgId": 1234567890123456
}
```

参数说明如表 6-98 所示。

表 6-98 参数说明

参 数	说 明
ToUserName	小程序的原始 ID
FromUserName	发送者的 openid
CreateTime	消息创建时间（整型）
MsgType	image
PicUrl	图片链接（由系统生成）
MediaId	图片消息媒体 id，可以调用获取临时素材接口拉取数据
MsgId	消息 id，64 位整型

- 进入会话事件。

用户在小程序“客服会话按钮”进入客服会话时将产生两种格式的数据包。

XML 格式的数据包：

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1482048670</CreateTime>
    <MsgType><![CDATA[event]]></MsgType>
    <Event><![CDATA[user_enter_tempsession]]></Event>
    <SessionFrom><![CDATA[sessionFrom]]></SessionFrom>
</xml>
```

json 格式的数据包：

```
{
    "ToUserName": "toUser",
    "FromUserName": "fromUser",
    "CreateTime": 1482048670,
    "MsgType": "event",
    "Event": "user_enter_tempsession",
```

```
    "SessionFrom": "sessionFrom"
}
```

参数说明如表 6-99 所示。

表 6-99　参数说明

参　数	说　明
ToUserName	小程序的原始 ID
FromUserName	发送者的 openid
CreateTime	事件创建时间（整型）
MsgType	event
Event	事件类型，user_enter_tempsession
SessionFrom	开发者在客服会话按钮设置的 sessionFrom 参数

2. 发送客服消息

当用户和小程序客服产生特定动作的交互时，微信会把消息数据推送给开发者，开发者可以在一段时间内（目前修改为 48 小时）调用客服接口，通过 POST 一个 json 数据包来发送消息给普通用户。此接口主要用于客服等有人工消息处理环节的功能，方便开发者为用户提供更加优质的服务。

目前允许的动作列表如表 6-100 所示，不同动作触发后，允许的客服接口下发消息条数和下发时限不同。下发条数达到上限后，会收到错误返回码，具体请见返回码说明页。

表 6-100　动作列表

用户动作	允许下发条数限制	下发时限
用户通过客服消息按钮进入会话	1 条	1 分钟
用户发送信息	3 条	48 小时

发消息的 HTTP 请求方式，POST 数据包如下：

```
https://api.weixin.qq.com/cgi-bin/message/custom/send?access_token=ACCESS_TOKEN
```

各消息类型所需的 json 数据包如下：

- 发送文本消息。

```
{
    "touser":"OPENID",
    "msgtype":"text",
    "text":
    {
        "content":"Hello World"
```

```
    }
}
```

- 发送图片消息。

```
{
    "touser":"OPENID",
    "msgtype":"image",
    "image":
    {
      "media_id":"MEDIA_ID"
    }
```

参数说明如表 6-101 所示。

表 6-101 参数说明

参 数	是否必须	说 明
access_token	是	调用接口凭证
touser	是	普通用户 openid
msgtype	是	消息类型，文本为 text，图片为 image
content	是	文本消息内容
media_id	是	发送图片的媒体 ID，通过新增素材接口上传图片文件获得

返回码说明如表 6-102 所示。

表 6-102 返回码说明

参 数	说 明
-1	系统繁忙，此时请开发者稍候再试
0	请求成功
40001	获取 access_token 时 AppSecret 错误，或者 access_token 无效。请开发者认真比对 AppSecret 的正确性，或查看是否正在为恰当的小程序调用接口
40002	不合法的凭证类型
4003	不合法的 openid，请开发者确认 openid 是否是其他小程序的 openid
45015	回复时间超过限制
45047	客服接口下行条数超过上限
48001	API 功能未授权，请确认小程序已获得该接口

3. 临时素材接口

小程序可以使用本接口获取客服消息内的临时素材（即下载临时的多媒体文件）。目前小程序仅支持下载图片文件。

接口调用的 HTTP 请求方式：GET、HTTPS 调用。

```
https://api.weixin.qq.com/cgi-bin/media/get?access_token=ACCESS_TOKEN&media_id=MEDIA_ID
```

【示例 6-72】

通过 curl 命令获取多媒体文件：

```
curl -I -G "https://api.weixin.qq.com/cgi-bin/media/get?access_ token=ACCESS_TOKEN&media_id=MEDIA_ID"
```

参数说明如表 6-103 所示。

表 6-103 参数说明

参　数	是否必须	说　明
access_token	是	调用接口凭证
media_id	是	媒体文件 ID

正确情况下返回 HTTP 头如下：

```
HTTP/1.1 200 OK
Connection: close
Content-Type: image/jpeg
Content-disposition: attachment; filename="MEDIA_ID.jpg"
Date: Sun, 06 Jan 2013 10:20:18 GMT
Cache-Control: no-cache, must-revalidate
Content-Length: 339721
curl -G "https://api.weixin.qq.com/cgi-bin/media/get?access_ token=ACCESS_TOKEN&media_id=MEDIA_ID"
```

如果返回的是视频消息素材，则内容如下：

```
{
 "video_url":DOWN_URL
}
```

错误情况下的返回 json 数据包示例如下（示例为无效媒体 ID 错误）：

```
{
  "errcode":40007,
  "errmsg":"invalid media_id"
}
```

小程序可以使用本接口把媒体文件（目前仅支持图片）上传到微信服务器，用户发送客服消息或被动回复用户消息。HTTP 请求方式：POST/FORM、HTTPS 调用。

```
https://api.weixin.qq.com/cgi-bin/media/upload?access_token=ACCESS_TOKEN
```

```
&type=TYPE
```

【示例 6-73】

使用 curl 命令，用 Form 表单方式上传一个多媒体文件：

```
curl -F media=@test.jpg "https://api.weixin.qq.com/cgi-bin/media/upload?access_token=ACCESS_TOKEN&type=TYPE"
```

参数说明如表 6-104 所示。

表 6-104 参数说明

参 数	是否必须	说 明
access_token	是	调用接口凭证
type	是	image
media	是	form-data 中媒体文件标识，有 filename、filelength、content-type 等信息

返回参数说明如表 6-105 所示。

表 6-105 返回参数说明

参 数	描 述
type	image
media_id	媒体文件上传后，获取标识
created_at	媒体文件上传时间戳

正确情况下的返回 json 数据包结果如下：

```
{
  "type":"TYPE",
  "media_id":"MEDIA_ID",
  "created_at":123456789
}
```

错误情况下的返回 json 数据包示例如下（示例为无效媒体类型错误）：

```
{
  "errcode":40004,
  "errmsg":"invalid media type"
}
```

4. 接入概述

接入微信小程序消息服务，开发者需要按照如下步骤完成：

- 填写服务器配置。
- 验证服务器地址的有效性。

- 依据接口文档实现业务逻辑。

【示例 6-74】

下面详细介绍这 3 个步骤。

- 填写服务器配置。

登录微信小程序官网后，在小程序官网的“设置→消息服务器”页面，管理员扫码启用消息服务，填写服务器地址（URL）、Token 和 EncodingAESKey。

URL 是开发者用来接收微信消息和事件的接口 URL。Token 可由开发者任意填写，用作生成签名（该 Token 会和接口 URL 中包含的 Token 进行比对，从而验证安全性）。 EncodingAESKey 由开发者手动填写或随机生成，将用作消息体加密、解密密钥。

开发者可选择的消息加、解密方式有：明文模式、兼容模式和安全模式。可以选择的消息数据格式：XML 格式或 json 格式。加密方式的默认状态是明文格式，而数据格式的默认状态是 XML 格式。

模式的选择与服务器配置在提交后都会立即生效，请开发者谨慎填写及选择。切换加密方式和数据格式需要提前配置好相关代码，详情请参考消息加、解密说明。

- 验证服务器地址的有效性。

开发者提交信息后，微信服务器将发送 GET 请求到填写的服务器地址 URL 上，GET 请求携带参数如表 6-106 所示。

表 6-106　GET请求携带参数参数

参　数	描　述
signature	微信加密签名，signature 结合了开发者填写的 Token 参数和请求中的 timestamp 参数、nonce 参数
timestamp	时间戳
nonce	随机数
echostr	随机字符串

开发者通过检验 signature 对请求进行校验。若确认此次 GET 请求来自微信服务器，请原样返回 echostr 参数内容，则接入生效，成为开发者，否则接入失败。加密/校验流程如下：

首先，将 Token、timestamp、nonce 三个参数进行字典序排序。

然后，将三个参数字符串拼接成一个字符串进行 sha1 加密。

最后，开发者获得加密后的字符串可与 signature 对比，表示该请求来源于微信。

检验 signature 的 PHP 示例代码：

```
private function checkSignature()
{
```

```
    $signature = $_GET["signature"];
    $timestamp = $_GET["timestamp"];
    $nonce = $_GET["nonce"];

    $token = TOKEN;
    $tmpArr = array($token, $timestamp, $nonce);
    sort($tmpArr, SORT_STRING);
    $tmpStr = implode( $tmpArr );
    $tmpStr = sha1( $tmpStr );

    if( $tmpStr == $signature ){
        return true;
    }else{
        return false;
    }
}
```

- 依据接口文档实现业务逻辑。

验证 URL 有效性成功后即接入生效，成为开发者。至此用户向小程序客服发送消息，或者进入会话等情况时，开发者填写的服务器配置 URL 将得到微信服务器推送过来的消息和事件，开发者可以依据自身业务逻辑进行响应。

注意：开发者所填写的 URL 必须以 http://或 https://开头，分别支持 80 端口和 443 端口。

6.9.6 分享

在 Page 中定义 onShareAppMessage 函数，设置该页面的分享信息。只有定义了此事件处理函数，右上角菜单才会显示“分享”按钮，用户单击“分享”按钮的时候会调用函数内容。此事件需要返回一个 Object，用于自定义分享内容。自定义分享字段如表 6-107 所示。

表 6-107 自定义分享字段

字 段	说 明	默 认 值
title	分享标题	当前小程序名称
desc	分享描述	当前小程序名称
path	分享路径	当前页面 path，必须是以/开头的完整路径

【示例 6-75】

```
Page({
  onShareAppMessage: function () {
    return {
      title: '自定义分享标题',
```

```
        desc: '自定义分享描述',
        path: '/page/user?id=123'
      }
    }
})
```

注意：用户进行分享的图片不能自行定义，系统会自动取得用户单击“分享”按钮时的当前页面。从此页面的顶部开始，选取高度为80%屏幕宽度的图形作为分享图片。

6.9.7 获取二维码

通过后台接口可以获取小程序任意页面的二维码，扫描该二维码可以直接进入小程序对应的页面，接口地址：

```
https://api.weixin.qq.com/cgi-bin/wxaapp/createwxaqrcode?access_token=ACCESS_TOKEN
```

POST参数说明如表6-108所示。

表6-108 POST参数说明

参　数	默认值	说　明
path		不能为空，最大长度128字节
width	430	二维码的宽度

【示例6-76】

获取二维码：

```
{"path": "pages/index?query=1", "width": 430}
```

注意：pages/index需要在app.json的pages中定义tip。通过该接口，仅能生成已发布的小程序的二维码。可以在开发者工具预览时生成开发版的带参二维码。带参二维码只有100000个，请谨慎调用。POST参数需要转成json字符串，不支持Form表单提交。

6.10 小结

本章介绍了微信小程序提供的API，并辅以示例进行分析，涵盖了几乎所有App常用功能提供接口的调用和方案。通过对于媒体、数据、设备、位置、界面设计，乃至支付，极大地丰富了小程序开发的拓展性和可用性。

到本章为止，小程序常用的API和技术均已介绍完毕，但是对于读者而言，真正的学习和练习可能才刚刚开始。微信小程序API极大地帮助了小程序开发者提高了开发速度，降低了难度，无须

底层调用和烦琐的写法，只要通过简单的 API 请求的方式，就可以实现想达到的功能。

6.11 实践与练习

其实对于微信小程序而言，最重要的不是如何去书写相关的代码，而是熟悉业务逻辑，精通流程设计。

- 读者可自行练习每一个需要调用的 API 接口。
- 读者可以通过不断地组合 API 获得更多更简单丰富的效果，并且可以找到其使用的环境。
- 读者可以认真思考每一种 API 在真实应用时的作用和效果，并可以分析其他现有 App 的构成。
- 可以在小程序中练习实现其他 App。

7 项目实战2——在线阅读

本章是一个完整的在线阅读实例，为了拓展读者的知识面，本章将会涉及后端语法和数据库使用。

在本章读者将会学到：

- 微信小程序的实践练习。
- 服务器端的基本逻辑和基础搭建。
- 数据库的基本逻辑和搭建。
- 服务器与微信小程序端的接口设计和交互逻辑。

注意：对于本书而言，后端服务器开发和数据库端并不是重点，如果读者需要更深入的学习，请参考相关图书。

7.1 项目准备工作

本章把案例作为一个完整的项目对待，所以在进行项目开发前，需要一些准备工作。

7.1.1 项目需求分析

本章指导读者开发一套小说阅读程序，程序会记录当前使用者的登录信息，具有书架功能，可以存储上次阅读过的书籍信息，并且能显示用户所有的书籍信息，单击即可进行图书的阅读。功能类似于当前流行的微信阅读、熊猫读书等 App。

本系统主要模拟了一个小程序实现的小说阅读功能，通过在线请求读取小说的一部分，可以通过单击按钮的上一页/下一页进行换页，或者使用直接输入页码的方式进行跳页。本系统还需要一个服务器端的管理后台，用于删除或者上传新的小说，形成了一个相对完整的项目。

根据上述需求，对系统进行如下总结：

1. 后端服务器部分

- 后端管理页面。
- 新小说的上传和填写简介等功能。
- 上传小说的管理。
- 所有用户管理。
- 提供小说翻页 API（前翻页、后翻页）。
- 提供小说的跳页 API（输入数字跳转相关页面）。
- 提供历史阅读和小说书签 API。
- 提供所有小说信息返回 API。
- 用户登录和注册 API。

2. 微信小程序需求

- 用户注册页面。
- 注册后的登录页面。
- 主页所有小说的显示。
- 点开相应小说历史阅读书签。
- 跳页和翻页功能。

7.1.2 项目系统设计

项目系统主要分为两部分：服务端、微信小程序的客户端。两者通过 HTTP 协议的 GET 和 POST 方式的 API 进行交互。基础构造如图 7-1 所示。



图 7-1 基础构造

7.1.3 设计 1：用户模块

对于用户模块，前端（用户端）的系统流程如图 7-2 所示。

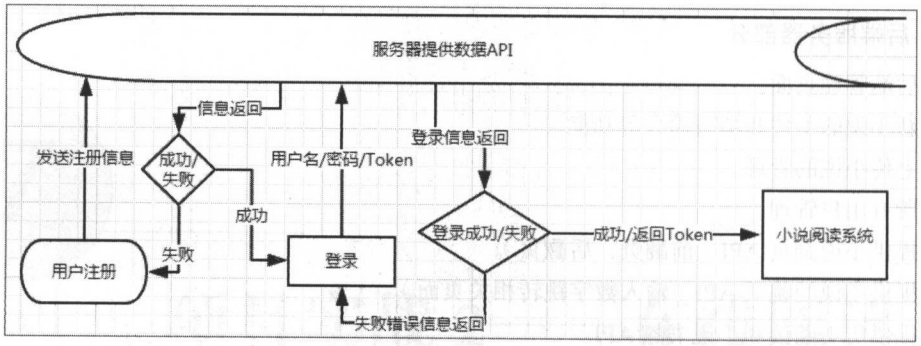


图 7-2　前端（用户端）的系统流程

服务器端，不同于微信服务端，需要直接和数据库进行交互，后台管理端同样也涉及了登录业务，所以其系统流程如图 7-3 所示。

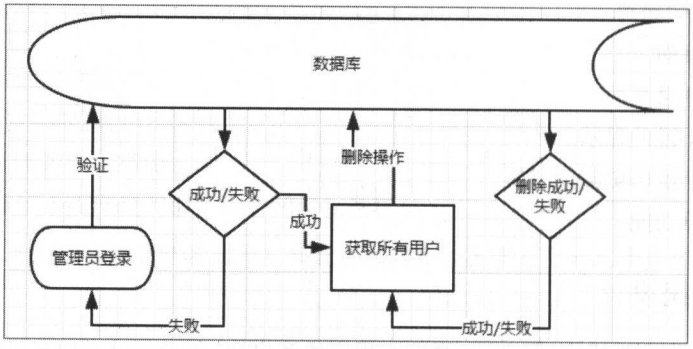


图 7-3　服务器端流程

当客户端发送请求后，后端服务器与微信小程序的具体交互部分，其服务器流程如图 7-4 所示。

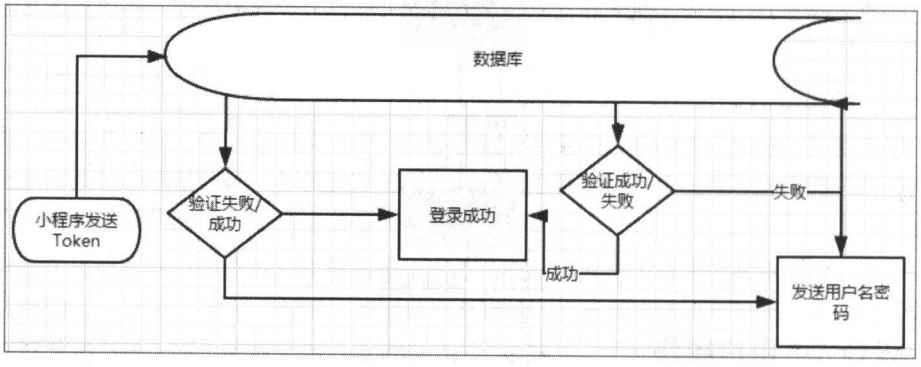


图 7-4　服务器流程

7.1.4 设计 2：小说阅读模块

小程序阅读模块本身也分为两个部分：针对应用使用者的客户端（微信小程序）部分，和服务器端部分。微信小程序端流程如图 7-5 所示。

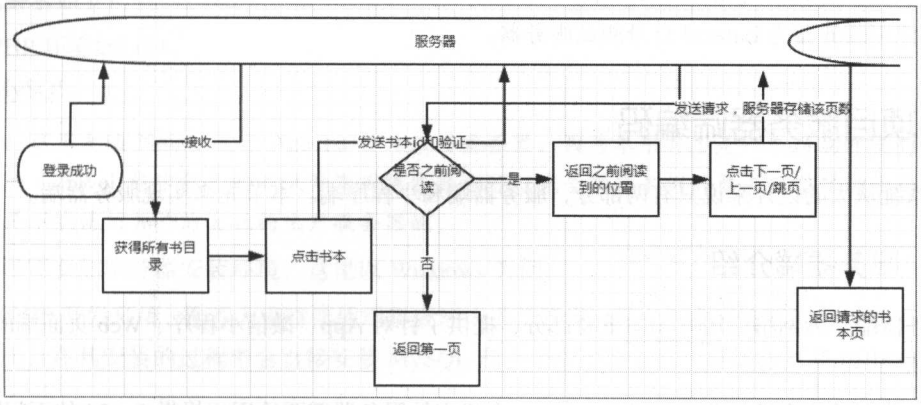


图 7-5 小程序端流程

服务器端的流程如图 7-6 所示。

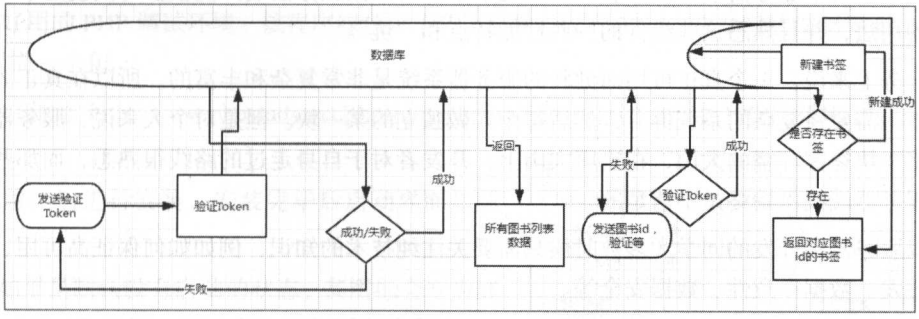


图 7-6 服务器端的流程

7.1.5 项目使用到的技术

本书作为一本微信小程序相关的书籍，对于服务器端的编码只会简单介绍，当然源码解析不能少。为了简化服务器端的开发，本例使用了 PHP 语言进行后端的支持。

注意：由于篇幅和精力有限，本文无法将一整套 Web 开发中的技术应用在本书的项目中，本项目只是为了让读者能快速掌握服务器端的开发，并且能更加深刻地理解 API 的交互本质。

本例采用以下技术：

- PHP 后端开发语言，使用开发框架 Laravel。
- MySQL 后端数据库。
- Bootstrap 和 jQuery 等作为前端框架和库。
- 微信开发 IDE 与相关开发技术。
- 使用服务器为 Laravel 自身测试服务器。

7.2 项目服务器端编码

本系统从大的设计来说只有两部分，服务器端和小程序端。本节先来实现服务器端。

7.2.1 服务器端介绍

服务器端作为网络应用开发的重要部分，提供了针对 App、微信小程序、Web 页面端的信息处理和存储层。

服务器端的稳定性是非常重要的一环，一个稳定的服务器需要给用户提供 7 × 24 的不间断服务，服务器端的开发经过了十多年的发展，而客户端的开发才刚刚兴起。所以服务器端的技术长久发展的结果就是，每个业务需求都已经有现成的轮子了。所以做服务器端开发很多时候就是学习各种开源组件的用法，并且熟悉这些组件的一些性能特点和“坑”。

从业务上来说，一个真正可用而健壮的服务器系统是非常复杂和丰富的。所以在真正的服务器端开发中，需要将复杂的系统拆分，然后每个人做独立的某一块。但是对个人来说，服务器端开发却始终是一块迷雾环绕的大山，在这座大山上，开发者对于自身走过的路线很熟悉，而那些没有接触过的服务器端技术始终是不清楚的。

所以服务器端开发的同学大多数时候只需要关注纯技术的知识，例如如何保证高可用、高扩展性、高并发、数据一致性、数据安全等。

本章将仅仅对于本例使用的服务器进行说明和分析，涉及的相应知识点可能会比较浅显，所以如果读者需要开发出更加稳定和健壮的系统，需要从其他途径进行相关的学习和延伸。

本服务器使用 PHP 进行开发，使用 MySQL 作为数据库，用于数据持久化存储，因为仅仅使用 API 提供的服务器，并且仅供测试，所以使用了 Laravel 框架进行开发和接口的编写。

7.2.2 Laravel 框架的安装与配置

Laravel 是一个开源的 PHP Web 框架，由 Taylor Otwell 创建，用于开发遵循模型-视图-控制器（MVC）架构模式的 Web 应用程序。Laravel 的一些特性是一个模块化包装系统，具有专用的依赖管理器，访问关系数据库的不同方法，有助于应用程序部署和维护实用程序，以及它对语法糖的定向。

截至 2015 年 3 月，Laravel 被认为是最流行的 PHP 框架之一，其源代码在 GitHub 上托管，并根据 MIT 许可证授权。

本系统服务器端使用的 Laravel 版本为 V5.3.*，它是本书写作时的最新版本，对于本例的服务器端环境要求如下：

- PHP 环境≥5.6.4。
- MySQL。

注意：这里无须 Nginx 或者 Apache 作为应用服务器，因为本书并不是一本后端开发的丛书，而在完整的 Laravel 包中包含的测试开发用服务器也可以完全符合本例的要求，当然对于正式上线的环境，是不能使用其作为真正的生产服务器的。

开发者需要的服务器安装环境，这里以 Windows 为例。

安装 PHP 开发环境，推荐大家直接使用集成环境进行开发，因为集成环境不仅仅只是一个 PHP 的环境，而且在其安装的过程中会直接安装 MySQL 和一个应用服务器，并且会打开 PHP 中对应的拓展。

这里使用 PHPStudy 进行环境的安装。该程序包集成了最新的 Apache+Nginx+LightTPD+PHP+MySQL+phpMyAdmin+Zend Optimizer+Zend Loader，一次性安装，无须配置即可使用，是非常方便、好用的 PHP 调试环境。该程序绿色、小巧、简易，仅有 32M，有专门的控制界面。总之学习 PHP 只需一个包。

（1）在地址 http://www.phpstudy.net/phpstudy/phpStudy20161103.zip 中下载最新的 PHPStudy 安装包。

（2）解压后进行安装，安装完毕后启动界面如图 7-7 所示。

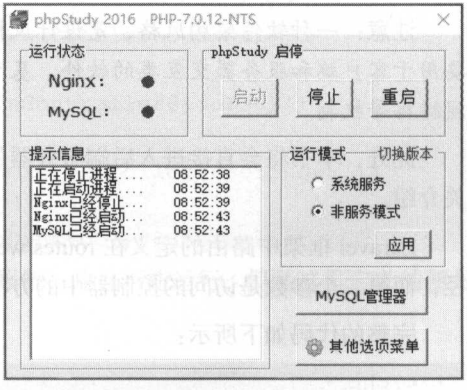


图 7-7 phpstudy 启动界面

（3）如果是默认没有启动的状态，其图标会变为红色，单击“启动”按钮，图标变为绿色则启动成功。

注意：这里需要选择相应的 PHP 版本，5.6 及以上。当无法启动 MySQL 时，请查看具体的错误命令，并且找出是由于端口占用或者其他情况导致的启动失败，然后根据错误信息重启服务。

（4）当启动成功后，使用浏览器输入 localhost，可以看到该 phpinfo 项目的初始化页面，使用命令行或 MySQL 工具可以进行 MySQL 的连接（其初始化的用户名和密码均为 root）。

注意：这里使用 Laravel 作为后端搭建框架，其开发环境和拓展要求为 PHP 版本≥5.6.4，PHP

扩展：OpenSSL，PHP 扩展：PDO，PHP 扩展：Mbstring 以及 PHP 扩展：Tokenizer。

（5）Laravel 使用 Composer 管理依赖，因此，使用 Laravel 之前，确保机器上已经安装了 Composer。

对于未安装 Composer 的机器，可以通过 win+R 组合件打开“运行”对话框，使用命令行进行安装：

```
php -r "readfile('https://getcomposer.org/installer');" | php
```

如果开启了 curl 可以使用：

```
curl -sS https://getcomposer.org/installer | php
```

注意：对于 Composer 的安装需要将 PHP 写入系统的环境变量，这点对于 Laravel 的使用也是必须的。

在命令行中输入 Composer 即可检查是否安装成功（需写入环境变量，且重启命令提示行）。

（6）安装 Laravel。使用通过 Composer 安装 Laravel 的安装器：composer global require "laravel/installer"（需写入系统路径，否则将不能在所有的目录下调用 Laravel）。

（7）提示安装成功后，输入 laravel new *命令建立新的项目。

注意：其他 Laravel 的相关命令和操作请读者自行查阅 Laravel 的官方文档以及相关的教程。

7.2.3 服务器功能路由划分

服务端现在流行的方案是 RESTful，而 Laravel 正是一个非常适用于开发此类风格的框架，开发过程的第一步就是确定提供 API 和页面访问的路由接口。

注意：一种软件架构风格，是设计风格而不是标准，只是提供了一组设计原则和约束条件，主要用于客户端和服务器交互类的软件。基于这个风格设计的软件可以更简洁、更有层次、更易于实现缓存等机制。

此处，本书将会直接进入后端服务器的开发，而不会在意一些技术细节，将直接给出代码和相关介绍。

Laravel 框架中路由的定义在 routes/web.php 中。其路由规则：第一个参数指定了相关的访问路径，而第二个参数是访问的控制器中的方法，使用 Route：post/get/……进行不同请求方法的划分。

完整的代码如下所示：

```
<?php

/*
 * 路由中的 API 为提供相关的 API 接口
 * Admin 为后台管理接口
```

```
 * */
//API 获取所有书的列表
Route::get('API/', 'Text\TextController@index');
//API 用户注册
Route::post('API/register', 'Text\TextController@register');
//API 获得用户 Token 用于之后的具体请求
Route::post('API/login', 'Text\TextController@getToken');
//API 获得个人书签和最近阅读
Route::post('API/books', 'Text\TextController@getUserBook');
//API 具体书本初始化阅读和下一页操作请求 API
Route::post('API/nextPage', 'Text\TextController@nextPage');
//书本上一页操作请求
Route::post('API/beforePage', 'Text\TextController@beforePage');
//书本跳转某页
Route::post('API/otherPage', 'Text\TextController@otherPage');
//管理登录界面
Route::get('/', 'Admin\AdminController@index');
//管理登录并且获得登录状态
Route::post('Admin/alogin', 'Admin\AdminController@login');
//管理界面主页
Route::get('Admin/index', 'Admin\AdminController@aIndex');
//管理书籍页面
Route::get('Admin/manger', 'Admin\AdminController@bookIndex');
//书籍管理删除方法
Route::get('Admin/bookDel', 'Admin\AdminController@bookDel');
//书籍管理，书籍上传
Route::post('Admin/addBook', 'Admin\AdminController@addBook');
//管理用户界面
Route::get('Admin/userManger', 'Admin\AdminController@userIndex');
//用户管理删除界面
Route::get('Admin/userDel', 'Admin\AdminController@userDel');
```

当然，此处并未定义相关的 Controller，所以代码会报错，并且不能运行，但是这里定义的便是本服务器所有可以提供功能的接口路径。

7.2.4 服务器数据库搭建

数据库在一个服务器端的程序中是非常重要的一部分，当然，本例的数据库只是一个小型应用而已。创建数据库时，要结合系统设计，搭建数据库的结构，这里不再赘述，直接给出数据库的设计方案。

本例只包含一个数据库，有四张相关的表。

（1）管理员用户表，命名为 admin_user，设计如图 7-8 所示。

名	类型	长度	十进位	允许空值 (	
id	int	11	0	☐	1
username	varchar	255	0	☐	
password	varchar	255	0	☐	

图 7-8　admin_user 表设计

（2）用户书签表，命名为 bookmarks，设计如图 7-9 所示。

名	类型	长度	十进位	允许空值 (	
id	int	11	0	☐	1
uid	int	11	0	☐	
bookid	int	11	0	☐	
page	int	11	0	☐	

图 7-9　bookmarks 表设计

（3）存储书本信息的表，命名为 books，设计如图 7-10 所示。

名	类型	长度	十进位	允许空值 (	
id	int	11	0	☐	1
name	varchar	255	0	☐	
src	varchar	255	0	☐	
time	int	11	0	☐	
number	int	11	0	☐	
writer	varchar	255	0	☐	
description	text	0	0	☑	

图 7-10　books 表设计

（4）用户表，命名为 users，设计如图 7-11 所示。

名	类型	长度	十进位	允许空值 (	
id	int	11	0	☐	1
time	int	11	0	☐	
username	varchar	20	0	☐	
password	varchar	255	0	☐	
token	varchar	255	0	☑	
nickname	varchar	10	0	☐	
ip	varchar	15	0	☐	

图 7-11　users 表设计

注意：上述所有的表，其 id 键都是自增形式的主键。

7.2.5 服务器的配置

首先是服务器的配置，配置文件在.env 中。

```
APP_ENV=local
APP_KEY=base64:BhNsLvDkLkuXHDnt4PN/MYeU8cVXRKW910L6D41QHXU=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=wx_demo_book
DB_USERNAME=root
DB_PASSWORD=root

BROADCAST_DRIVER=log
CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

PUSHER_APP_ID=
PUSHER_KEY=
PUSHER_SECRET=
```

这里主要介绍一般使用的几个字段：

- App_DEBUG：项目 debug 配置，配置错误输出，对于基本的修改会自动清理缓存。
- DB_CONNECTION：数据库连接方式，这里选择 MySQL。
- DB_HOST：数据库连接地址 IP，这里是本机的 MySQL 库。

- DB_ PORT：数据库连接端口，这里使用 MySQL 基本的 3306。
- DB_DATABASE：数据库名称。
- DB_USERNAME：数据库用户名称。
- DB_PASSWORD：数据库用户密码。

注意：在运行本例的时候，框架将会自动载入此配置文件，并对配置进行修改。

通过此配置，程序会自动连接到相关的数据库，然后可以对数据库进行查找、添加、删除等操作，也相当于增加了一个数据持久层。

7.2.6 服务器的项目结构

服务器的整个项目结构如图 7-12 所示。

其中，Controllers 文件夹作为控制器存在，所有的页面逻辑都在其中，本项目分为两部分：Admin 文件夹中是网站后台管理的逻辑；提供小程序使用的 API，位于 Text 文件夹。

对于后台管理页面的代码，根据路由设计，通过路由提供的方法实现，使用$request 获得前台和输入请求的值，并且对基本的用户名和密码进行验证。

注意：这里保留的密码为明文，这种方式是不安全且不合适的行为，不能在正式的服务器上使用。

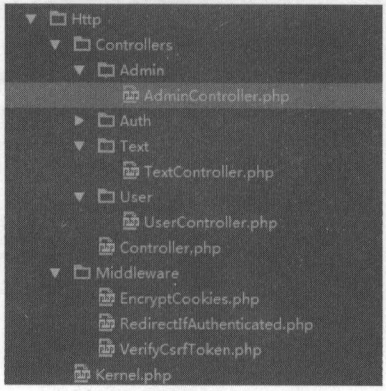


图 7-12 项目结构

7.2.7 服务器端登录页面的代码实现

使用者进入管理页面，输入正确的用户名和密码，验证成功后会自动跳转至管理的主页，并且显示主页。

完整的 Admin 代码如下：

```
<?php
namespace App\Http\Controllers\Admin;

use DB;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Storage;

class AdminController extends Controller
{
```

```
    public function index()
    {
        return view('Admin/login');
    }

    public function login(Request $request)
    {
        $username = $request->input('username');
        $password = $request->input('password');
        if ($username != null && $password != null) {
            $user       =       DB::table('admin_user')->where('username',
$username)->where('password', $password)->first();
            if ($user) {
                $request->session()->put('user', $user);
                return redirect('Admin/index');
            } else {
                return view('Public/error', ['title' => "密码错误", 'message'
=> "就是你输错密码了"]);
            }
        } else {
            return view('Public/error', ['title' => "输入为空", 'message' =>
"不要输空的用户名和密码"]);
        }
    }

    //主页显示
    public function aIndex(Request $request)
    {
        if($this->checkUserIsLogin($request)) {
            return view('Admin/index');
        }else{
            return redirect('../');
        }
    }

}
```

当然了，除了登录页面以外，所有的页面都需要在登录后才能进入，这里只需要一个简单的方法来判定存储的用户信息，就可以进行一个简单的权限控制操作，对于非法进入的执行跳转操作。

```
    //检测是否登录
    public function checkUserIsLogin(Request $request)
```

```
    {
        //检测是否登录
        $value = $request->session()->get('user');
        if($value!=''){
            return true;
        }else{
            return false;
        }
    }
```

7.2.8 服务器端管理主页的代码实现

本系统管理主页实现的功能为书籍的上传、删除，通过数据库的读取，获得已经添加过的书籍的数据和基础的历史添加书籍信息，还可以对数据进行管理。

其中，最重要的是实现管理员对书籍的上传操作，通过浏览器上传 txt 文件的形式，进行文件的存储，并且通过生成唯一的书籍名称来防止上传的冲突。其逻辑代码如下：

```
//主页添加书籍
    public function addBook(Request $request)
    {
        if($this->checkUserIsLogin($request)) {
            //判断请求中是否包含 name=file 的上传文件
            if (!$request->hasFile('file')) {
                exit('上传文件为空! ');
            }
            $file = $request->file('file');
            //判断文件上传过程中是否出错
            if (!$file->isValid()) {
                exit('文件上传出错! ');
            }
            $destPath = realpath(public_path('books'));
            if (!file_exists($destPath))
                mkdir($destPath, 0755, true);
            //重新更新名称，md5()加密生成唯一名称
            $filename = md5(($file->getClientOriginalName()) . time()) .
'.txt';
            if (!$file->move($destPath, $filename)) {
                exit('保存文件失败! ');
            }
            //获取所有的数据
```

```
            $data = $request->all();
            $data_temp = array(
                'name' => $data['name'],
                'src' => '/books/' . $filename,
                'time' => 0,
                'number' => 0,
                'writer' => $data['writer'],
                'description' => $data['description']
            );
            //获取 file，并调用 putFile 进行，写入 src
            //构造 data 为符合 DB 要求的变量
            if (DB::table('books')->insert($data_temp)) {
                return redirect('Admin/index');
            } else {
                return view('Public/error', ['title' => "存储错误", 'message'
=> "可能输入或者是数据库错误"]);
            }
        }else{
            return redirect('../');
        }

    }

    //书籍管理主页显示
    public function bookIndex(Request $request)
    {
        if($this->checkUserIsLogin($request)) {
            $data = DB::table('books')->get();
            return view('Admin/bookIndex', ['books' => $data]);
        }else{
            return redirect('../');
        }
    }

    //书籍管理，书籍删除
    public function bookDel(Request $request)
    {
        if($this->checkUserIsLogin($request)) {
            $id = $request->input('bookId');
            DB::table('books')->where('id', '=', $id)->delete();
            return redirect('Admin/manger');
```

```
        }else{
            return redirect('../');
        }
    }
```

7.2.9 服务器端管理用户的代码实现

在此页面对使用者的账号进行查阅和管理，当然，也可以进行其他用户操作的扩充。

```
//用户管理主页显示
    public function userIndex(Request $request)
    {
        if($this->checkUserIsLogin($request)) {
            $data = DB::table('users')->get();
            return view('Admin/userIndex', ['users' => $data]);
        }else{
            return redirect('../');
        }
    }

    //用户管理主页显示
    public function userDel(Request $request)
    {
        if($this->checkUserIsLogin($request)){
            $id = $request->input('userId');
            DB::table('users')->where('id', '=', $id)->delete();
            return redirect('Admin/userManger');
        }else{
            return redirect('../');
        }
    }
```

7.2.10 小程序的接口文档

小程序的接口文档基本实现的原理是对一本已经上传到服务器的 txt 文档书籍进行切分，收到请求后进行字段的切分，从而形成简单的分页系统。

注意：这里并非小说网站实现的手法，仅仅是作为一个测试展示的 Demo，实现方式非常简单且不存在延迟。这种方法也有很大的局限性，就是对于文字编码方式的要求，上传的所有文章需要先对 txt 的编码进行 UTF-8 转码，且中文在切割后可能会产生部分文字消失的情况。

提供小程序使用的接口文件的代码如下：

```
<?php
namespace App\Http\Controllers\Text;

use DB;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class TextController extends Controller
{
//格式化返回值
    public function formatData($status, $message, $data)
    {
        $temp = array(
            'status' => $status,
            'message' => $message,
            'data' => $data
        );
        return json_encode($temp);
    }

    //在每次调用 API 的时候进行一次 session 检测，确保每次登录后使用 API 接口
    public function checkUser($token)
    {
        //检测 Session 但是不检测其是否过期等
        $user = DB::table('users')->where('token', $token)->first();
        if ($user) {
            return $user;
        } else {
            return false;
        }
    }

}
```

（1）首先是对于用户登录的统一检测页面和对于数据的统一封装，然后返回给客户端的方法。通过用户发送的令牌值，来获得是否是正确的用户和正确的请求数据。

（2）第二部分是用户的登录和注册，并对用户的 Token 进行一个有效期的验证，这样的验证方式也防止因为长时间的用户信息记录而导致的用户个人信息泄露情况。具体的代码如下：

```
//注册
    public function register(Request $request)
    {
```

```
            //email 中的 username
            $username = $request->get('username');
            $password = $request->get('password');
            $nickname = $request->get('nickname');
            if (DB::tables('users')->where('username', $username)-> first()) {
                return $this->formatData(0, "您似乎已经注册过了，请直接登录", null);
            } else {
                $data = array(
                    'username' => $username,
                    'password' => $password,
                    'nickname' => $nickname,
    'time'=>time(),
                    'ip'=>'0.0.0.0',

                );
                if (DB::table('users')->insert($data)) {
                    return $this->formatData(1, null, null);
                } else {
                    return view('Public/error', ['title' => "存储错误", 'message'
=> "可能输入或者是数据库错误"]);
                }
            }
        }

        //用户登录获取 Token
        function getToken(Request $request)
        {
            //此处 Token 使用时间戳和用户名之后使用 Base64 生成
            $username = $request->input('username');
            $password = $request->input('password');
            $token = $request->input('token');
            if ($token == '') {
                if ($username == '' || $password == '') {
                    //做用户信息的完整验证
                    return $this->formatData(0, '请输入合适的用户名和密码', null);
                } else {
                    //验证用户名与密码
                    $user        =        DB::table('users')->where('username',
$username)->first();
                    if ($user->password == $password) {
                        //生成新 Token
```

```
                    $token = base64_encode(time() . md5($username));
                    //存储新 Token
                    DB::table('users')->where('id',
$user->id)->update(['token' => $token]);
                    return  $this->formatData(1,  null,  array('token'  =>
$token));
                } else {
                    return $this->formatData(0, "输入的密码或者用户名出错",
null);
                }
            }
        } else {
            //验证 Token 是否过期
            $token_decode = base64_decode($token);
            if ($token_decode) {
                $user = DB::table('users')->where('token', $token)->first();
                if ($user) {
                    //已经验证成功，验证是否过期，切分字符串，取到相关数字
                    $time = substr($token_decode, 0, (mb_strlen($token_decode)
- mb_strlen(md5($user->username))));
                    //设置 15 天过期
                    if (((int)$time + 1296000) > time()) {
                        //写入相应的 session 供其他的接口检测而不用查数据库
                        return $this->formatData(1, '自动登录成功', null);
                    } else {
                        return $this->formatData(0, '您的登录已经过期，请输入使用
的用户名和密码', null);
                    }
                } else {
                    return $this->formatData(0, '您的自动登录不合法，请输入合适的
用户名和密码', null);
                }
            } else {
                return $this->formatData(0, '您的登录出错，请输入合适的用户名和密
码', null);
            }
        }
    }
```

（3）第三部分是对已登录用户之前阅读的书本信息的获取，还有所有书本的获取。已阅读书本的获取，代码如下：

```
    //获得用户之前阅读的书与阅读书的书签
    public function getUserBook(Request $request)
    {
        $token = $request->input('token');
        $user = $this->checkUser($token);
        if ($user) {
            //取得书和页数列表
            $books_my = DB::table('bookmarks')->where('uid',
$user->id)->select('bookid')->get();
//          var_dump($books_my);die;
            //循环查找到的阅读历史，在books表中查找相关的知识点
            foreach ($books_my as $k => $v) {
                $books[$k] = DB::table('books')->where('id',
$v->bookid)->get();
            }
//          var_dump($books);die;
            return $this->formatData(1, null, $books);
        } else {
            return $this->formatData(0, "您的登录似乎存在问题，获取失败", null);
        }
    }
    //获得添加的所有图书列表
    public function index()
    {
        $books = DB::table('books')->get();
        if ($books) {
            return $this->formatData(1, null, $books);
        } else {
            return $this->formatData(0, '请联系管理员', null);
        }
    }
```

对所有书本的划分和基础的跳页功能，使用一些参数标识量来区分前进一页，和后退一页的操作。

```
    //将读取全部变成统一方法，$status用于-1、1和0，当0时$page或者是具体的跳页数字
    function bookCut(Request $request, $status, $page)
    {
        $bookId = $request->input('bookId');
        $token = $request->input('token');
        $user = $this->checkUser($token);
        if ($user) {
```

```
            //读取数据库中的书本和其 src
            $book = DB::table('books')->where('id', $bookId)->first();
            if ($book) {
                $books_my       =       DB::table('bookmarks')->where('uid',
$user->id)->where('bookid', $bookId)->first();
                if ($books_my) {
                    //已经存在的书签和标记，更新页码
                    //-1 时上一页
                    if ($status == -1) {
                        //已经存在的书签和标记，更新页码（减少一页）
                        DB::table('bookmarks')->where('uid',
$user->id)->where('bookid', $bookId)->update(['page' => $books_my->page - 1]);
                        $page = $books_my->page - 1;
                    } else if ($status == 1) {
                        //+1 时下一页
                        DB::table('bookmarks')->where('uid',
$user->id)->where('bookid', $bookId)->update(['page' => $books_my->page + 1]);
                        $page = $books_my->page;
                    } else if ($status == 0) {
                        //直接跳转到该页
                        //记录之前的 page，用于回滚 page
                        $page_old = $books_my->page;
                        //已经存在的书签和标记，更新页码（减少一页）
                        DB::table('bookmarks')->where('uid',
$user->id)->where('bookid', $bookId)->update(['page' => $page]);
                    }
                } else {
                    //建立新的书签和标记(未读的流程)
                    DB::table('bookmarks')->insert(['uid'    =>    $user->id,
'bookid' => $bookId, 'page' => 1]);
                    $page = 0;
                }
                //读取 txt 文件
                $data = file_get_contents(public_path() . $book->src, NULL,
NULL, $page * 1000, 1000);
                if ($data == '') {
                    if ($status == -1) {
                        //上一页的操作
                        //当已经达到第 0 页时，回滚一页
                        DB::table('bookmarks')->where('uid',
```

```
$user->id)->where('bookid', $bookId)->update(['page' => $books_my->page + 1]);
                    return $this->formatData(0, "您已读完此书", null);
                } elseif ($status == 1) {
                    //当下一页已经是
                    //当已经读完此书时，回滚一页
                    DB::table('bookmarks')->where('uid',
$user->id)->where('bookid', $bookId)->update(['page' => $books_my->page]);
                    return $this->formatData(0, "您已读完此书", null);
                } elseif ($status == 0) {
                    //跳出不符合范围的
                    //当已经读完此书时，或者跳到了一些奇怪的位置时，回滚原来的页
                    DB::table('bookmarks')->where('uid',
$user->id)->where('bookid', $bookId)->update(['page' => $page_old]);
                    return $this->formatData(0, "您已经跳出页码范围", null);
                }
            } else {
                $data = mb_convert_encoding($data, 'utf-8');
                return $this->formatData(1, null, $data);
            }
        } else {
            return $this->formatData(0, "您需要看的书不存在", null);
        }
    } else {
        return $this->formatData(0, "您的登录似乎存在问题，获取失败", null);
    }
}

//下一页
function nextPage(Request $request)
{
    $data=$this->bookCut($request, 1, 0);
    return $data;
}

//上一页
function beforePage(Request $request)
{
    $data=$this->bookCut($request, -1, 0);
    return $data;
}
```

```
    //跳转页面
    function otherPage(Request $request){
        $page=$request->input('page');
        $data=$this->bookCut($request,0,$page);
        return $data;
    }
```

（4）最后一部分是关于用户对自身阅读的历史及书签的删除操作，通过此接口可以对书签进行删除。

```
    //用户对于之前阅读书的删除
    public function delUserBook(Request $request)
    {
        $token = $request->input('token');
        $user = $this->checkUser($token);
        if ($user) {
            //取得书和页数列表
            $bookId = $request->input('bookId');
            DB::table('bookmarks')->where('uid',
$user->id)->where('bookid', $bookId)->delete();
            return $this->formatData(1, "删除成功", null);
        } else {
            return $this->formatData(0, "您的登录似乎存在问题，删除失败", null);
        }
    }
```

7.2.11 后台管理页面搭建

文章和用户的后台管理，需要一个简单的登录页面和基础的管理页面，这就用到了 Laravel 框架，其模板地址为固定模块的 views 文件夹。本测试服务器的文件结构如图 7-13 所示。

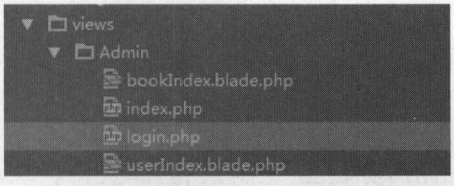


图 7-13 文件结构

（1）首先是一个 Login 页面，即用户在打开此管理界面如果未登录而自动跳转到登录界面，其代码如下所示。

```
<!DOCTYPE html>
<html>
<head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> 登录</title>
    <script>if(window.top !== window.self){ window.top.location = window.location;}</script>
</head>

<body class="gray-bg">

    <div class="middle-box text-center loginscreen animated fadeInDown">
        <div>
            <div>
                <h1 class="logo-name">ST</h1>
            </div>
            <h3>欢迎使用管理上传功能</h3>
            <form class="m-t" role="form" METHOD="POST" action="Admin/alogin">
                <div class="form-group">
                    <input type="text" name="username" class="form-control" placeholder="用户名" required="">
                </div>
                <div class="form-group">
                    <input type="password" name="password" class="form-control" placeholder="密码" required="">
                </div>
                <input type="hidden" name="_token" value="<?php echo csrf_token()?>"/>
                <button type="submit" class="btn btn-primary block full-width m-b">登 录</button>
                </p>
            </form>
        </div>
    </div>
    <script src="admin/js/jquery.min.js?v=2.1.4"></script>
    <script src="admin/js/bootstrap.min.js?v=3.3.6"></script>
</body>

</html>
```

注意：如果读者在实现的时候，对于某些 JS 文件或者 css 文件出现错误时，只要能实现效果，无须担心，本测试服务器只是为了能达到可用的效果，而非要求完美的样式和动态效果。

（2）对于 bookIndex，其主要实现的是文章的管理主页，为了方便起见，只是实现了一个删除功能。将所有的书本信息进行列表式展示，并且在每一本不同的书的一行，提供简单的管理操作按钮。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
    <!-- The above 3 meta tags *must* come first in the head; any other head
content must come *after* these tags -->
    <meta name="description" content="">
    <meta name="author" content="">

    <title>主页</title>

    <!-- Bootstrap core CSS -->
    <link
href="//cdn.bootcss.com/bootstrap/4.0.0-alpha.5/css/bootstrap.min.css"
rel="stylesheet">

</head>

<body>

<div class="container">
    <div class="header clearfix">
        <nav>
            <ul class="nav nav-pills float-xs-right">
                <li class="nav-item">
                    <a class="nav-link" href="index">书籍上传 <span
class="sr-only">(current)</span></a>
                </li>
                <li class="nav-item">
                    <a class="nav-link active" href="manger">书籍管理</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="userManger">用户管理</a>
```

```
                </li>
            </ul>
        </nav>
    </div>
    <div class="row marketing" style="padding-top: 10%">
        <div class="col-lg-12">
            <table class="table">
                <thead class="thead-default">
                <tr>
                    <th>#</th>
                    <th>名称</th>
                    <th>地址</th>
                    <th>阅读次数</th>
                    <th>number</th>
                    <th>作者</th>
                    <th>操作</th>
                </tr>
                </thead>
                <tbody>
                @foreach ($books as $book)
                    <tr>
                        <th>{{$book->id}}</th>
                        <th>{{$book->name}}</th>
                        <th>{{$book->src}}</th>
                        <th>{{$book->time}}</th>
                        <th>{{$book->number}}</th>
                        <th>{{$book->writer}}</th>
                        <th>
                            <a href=""><button class="btn btn-warning">修改
</button></a>
                            <a    href="bookDel?bookId={{$book->id}}"><button
class="btn btn-danger">删除</button></a>
                        </th>
                    </tr>
                @endforeach
                </tbody>
            </table>
        </div>
    </div>

    <footer class="footer">
```

```
        <p>&copy; 2016</p>
      </footer>

   </div>
   </body>
   </html>
```

（3）文章后台管理的 index.php 主页，主要实现对于书籍信息上传操作的页面。在此页面存在一个上传书本信息的表单。

```
   <!DOCTYPE html>
   <html lang="en">
   <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
      <!-- The above 3 meta tags *must* come first in the head; any other head
content must come *after* these tags -->
      <meta name="description" content="">
      <meta name="author" content="">

      <title>主页</title>

      <!-- Bootstrap core CSS -->
      <link
href="//cdn.bootcss.com/bootstrap/4.0.0-alpha.5/css/bootstrap.min.css"
rel="stylesheet">

   </head>

   <body>

   <div class="container">
      <div class="header clearfix">
         <nav>
            <ul class="nav nav-pills float-xs-right">
               <li class="nav-item">
                  <a class="nav-link active" href="index">书籍上传<span
class="sr-only">(current)</span></a>
               </li>
               <li class="nav-item">
```

```
                    <a class="nav-link" href="manger">书籍管理</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="userManger">用户管理</a>
                </li>
            </ul>
        </nav>
    </div>
    <div class="row marketing" style="padding-top: 10%">
        <div class="col-lg-12">
            <form      action="addBook"      enctype="multipart/form-data"
method="post">
                <fieldset class="form-group">
                    <div class="form-group row">
                        <label class="form-control-label">书名: </label>
                        <input class="form-control" type="text" name="name">
                    </div>
                    <div class="form-group row">
                        <label class="form-control-label">作者: </label>
                        <input class="form-control" type="text" name="writer">
                    </div>
                    <div class="form-group row">
                        <label class="form-control-label">描述: </label>
                        <textarea style="height: 80px" class="form-control"
type="text" name="description"></textarea>
                    </div>
                    <div class="form-group row">
                        <label class="form-control-label">文件: </label>
                        <input class="form-control" type="file" name="file">
                    </div>
                    <input type="hidden" name="_token" value="<?php echo
csrf_token()?>"/>
                    <div class="form-group row">
                        <input class="btn btn-secondary" type="submit" value="
提交" name="submit">
                    </div>
                </fieldset>
            </form>
        </div>
    </div>
```

```
      <footer class="footer">
         <p>&copy; 2016</p>
      </footer>

   </div>
   </body>
   </html>
```

（4）用户管理页面 userIndex，在本页面使用列表的形式将所有的使用者资料列出，还可以对每一行的数据进行简单的管理操作。

```
   <!DOCTYPE html>
   <html lang="en">
   <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta  name="viewport"  content="width=device-width,  initial-scale=1,
shrink-to-fit=no">
      <!-- The above 3 meta tags *must* come first in the head; any other head
content must come *after* these tags -->
      <meta name="description" content="">
      <meta name="author" content="">

      <title>主页</title>

      <!-- Bootstrap core CSS -->
      <link  href="//cdn.bootcss.com/bootstrap/4.0.0-alpha.5/css/bootstrap.
min.css" rel="stylesheet">

   </head>

   <body>

   <div class="container">
      <div class="header clearfix">
         <nav>
            <ul class="nav nav-pills float-xs-right">
               <li class="nav-item">
                  <a class="nav-link" href="index">书籍上传<span class="sr-
only">(current)</span></a>
               </li>
               <li class="nav-item">
```

```
                    <a class="nav-link" href="manger">书籍管理</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link active" href="userManger">用户管理</a>
                </li>
            </ul>
        </nav>
    </div>
    <div class="row marketing" style="padding-top: 10%">
        <div class="col-lg-12">
            <table class="table">
                <thead class="thead-default">
                <tr>
                    <th>#</th>
                    <th>最近登录时间</th>
                    <th>用户名</th>
                    <th>昵称</th>
                    <th>最近登录 ip</th>
                    <th>操作</th>
                </tr>
                </thead>
                <tbody>
                @foreach ($users as $user)
                    <tr>
                        <th>{{$user->id}}</th>
                        <th>{{$user->time}}</th>
                        <th>{{$user->username}}</th>
                        <th>{{$user->nickname}}</th>
                        <th>{{$user->ip}}</th>
                        <th>
                            <a href=""><button class="btn btn-warning">修改
</button></a>
                            <a href="userDel?userId={{$user->id}}"> <button
class="btn btn-danger">删除</button></a>
                        </th>
                    </tr>
                @endforeach
                </tbody>
            </table>
        </div>
    </div>
```

```
    <footer class="footer">
        <p>&copy; 2016</p>
    </footer>

</div>
</body>
</html>
```

（5）这里还自定义了一个报错页面，用于当用户访问不允许访问的目录或者是做一些非法访问的时候进行页面的跳转。

```
<html>
<head>
    <title><?php echo $title;?></title>
</head>
<body>
<div style="padding-top: 20%;padding-left: 40%">
    <img src="../public/error.gif" style="width:30%;">
    <h3><?php echo $message;?><a href="javascript:history.go(-1)">返回上一
页</a> </h3>
</div>

</body>
</html>
```

7.2.12 服务器端的测试

本例测试服务器使用 Laravel 自带的开发测试服务器，在命令行中输入以下命令可以启动测试服务器。

```
php artisan serve
```

运行效果如图 7-14 所示。此服务器会自动启动本地的 8000 端口，所以访问 http://localhost:8000 即可访问本网站。

```
E:\PHP\WX_demo_read2>php artisan serve
Laravel development server started on http://localhost:8000/
```

图 7-14 运行效果

注意：使用此测试服务器时不能将命令行窗口关闭，或使用 Ctrl+c 组合键结束此命令，否则将视为关闭此服务器。同时，用户的访问信息、请求信息，以及出现的错误信息都将打印在此命令行上。

（1）进入此地址，首先会登录界面，其界面如图 7-15 所示。

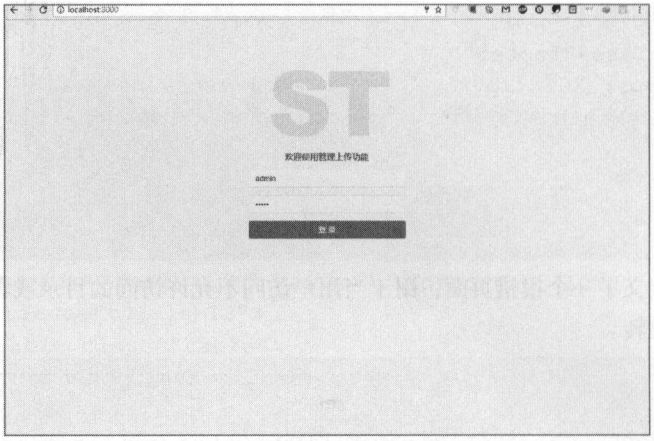


图 7-15　前台登录界面

（2）输入正确的用户名和密码后，单击“登录”按钮，可以跳转至后台的主页，此时，后台的主页是书籍的上传页，如图 7-16 所示。



图 7-16　书籍的上传页

（3）单击右上方的导航按钮“书籍管理”，可以进入书籍管理页面，此时数据库中存在一个数据内容，如图 7-17 所示。

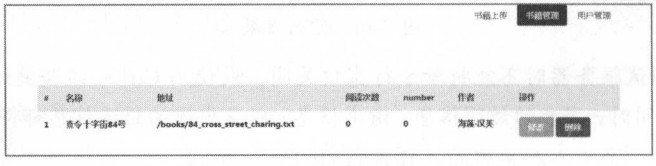


图 7-17　书籍管理页面

（4）单击“用户管理”导航按钮，可以进入用户导航页面，这里将会显示所有的用户和对用户的管理操作，如图 7-18 所示。

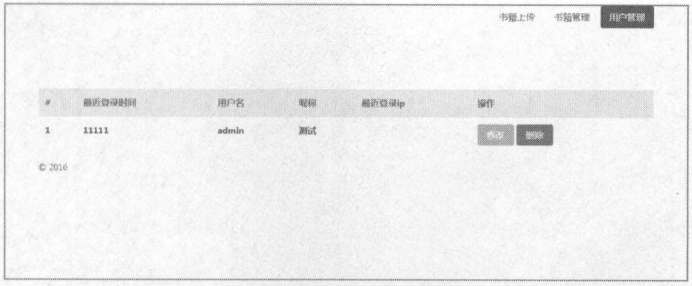


图 7-18 用户管理页面

（5）对于 API 接口的验证，不能简单使用界面来验证，要使用一些第三方的工具，这里使用 postman 进行接口的验证。

注意：postman 插件充当着这种交互方式的“桥梁”，它可以利用 Chrome 插件的形式把各种模拟用户 HTTP 请求的数据发送到服务器，以便开发人员能够及时地作出正确的响应，或者是对产品发布之前的错误信息提前处理，进而保证产品上线之后的稳定性和安全性。

首先访问的是不需要验证的路由地址/API/，其主要是为了获得所有的书籍信息，完整的数据接口地址为 http://localhost:8000/API/，使用了 GET 的方式进行获取。获取情况如图 7-19 所示。

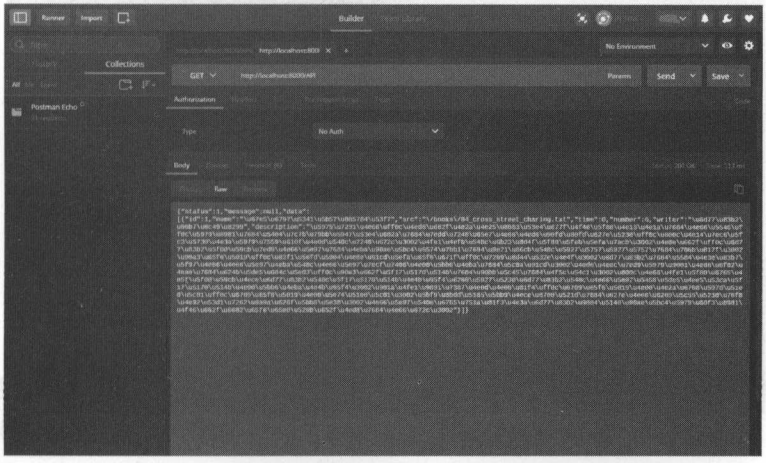


图 7-19 获取书籍信息

测试用户的登录信息，输入用户名和密码然后获取 Token，测试地址为 http://localhost:8000/API/login，需要选择获取方式为 POST，然后在 body 中传入 username 和 password。

如果输入正确，将会返回对应的 userToken 值，否则将会返回错误标识和错误信息，如图 7-20 所示。

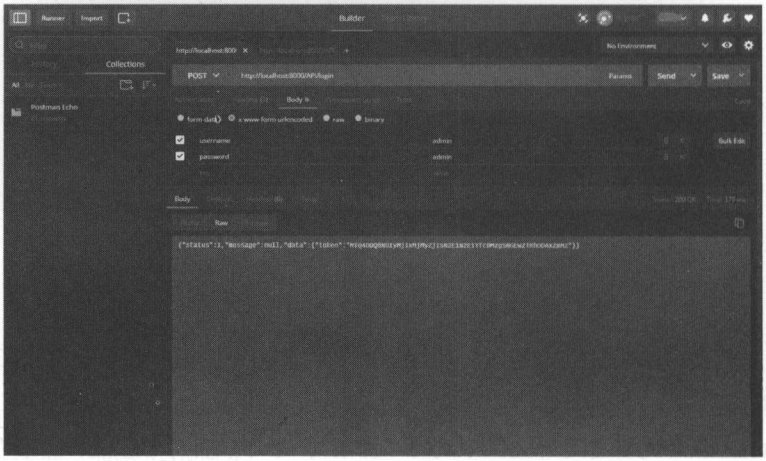


图 7-20　登录成功后

测试对用户阅读历史的获取，使用 POST 方式，在表单中需要传入用户的 Token 作为验证。其访问地址为 http://localhost:8000/API/books，访问成功后返回书本信息，如图 7-21 所示。

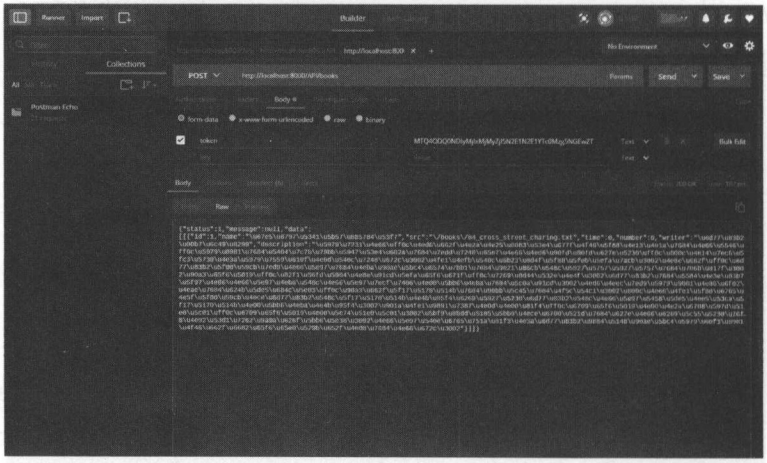


图 7-21　成功后返回书本信息

对书本进行测试，使用的测试地址为 http://localhost:8000/API/nextPage，要传入用户 Token 用作验证和书本的 bookId。

这里选择之前阅读过相同的书本时会自动跳转至之前阅读的页数，具体显示效果如图 7-22 所示。

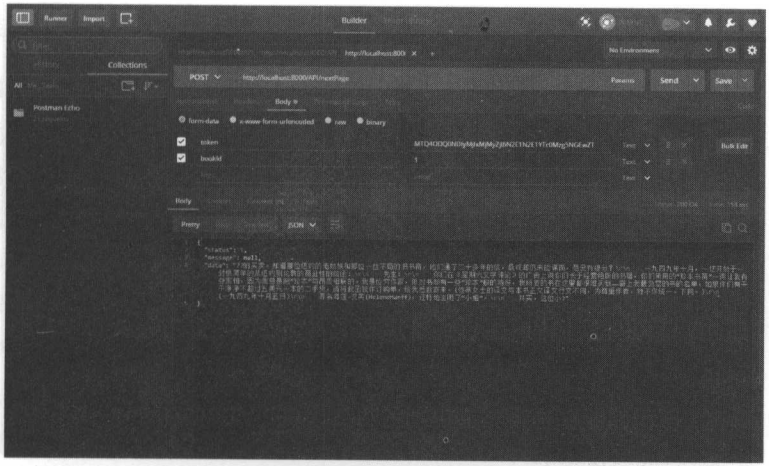


图 7-22 请求书本内容

由于书本信息的初始化其实是相当于阅读了第一页，所以和翻页接口写成一个相同的接口，当有下一次的请求时会实现自动翻页。图 7-23 为再次请求后的翻页效果。

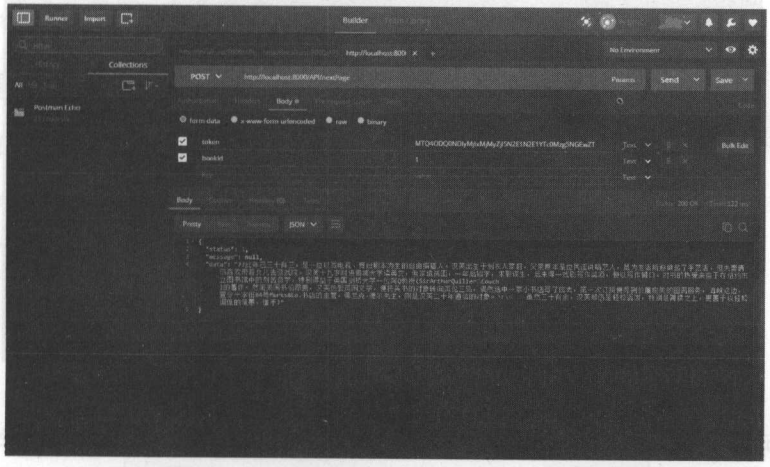


图 7-23 再次请求后的翻页效果

同样，本系统提供了上一页功能，其请求地址为 http://localhost:8000/API/beforePage，当请求此地址时，将会返回上一页，显示效果如图 7-24 所示。

为了方便读者使用本系统，在翻页功能的基础上增加了跳页的功能，需要传递一个新的参数 page，请求地址为 http://localhost:8000/API/otherPage ，这里选择了 page=20 的情况，显示效果如图 7-25 所示。

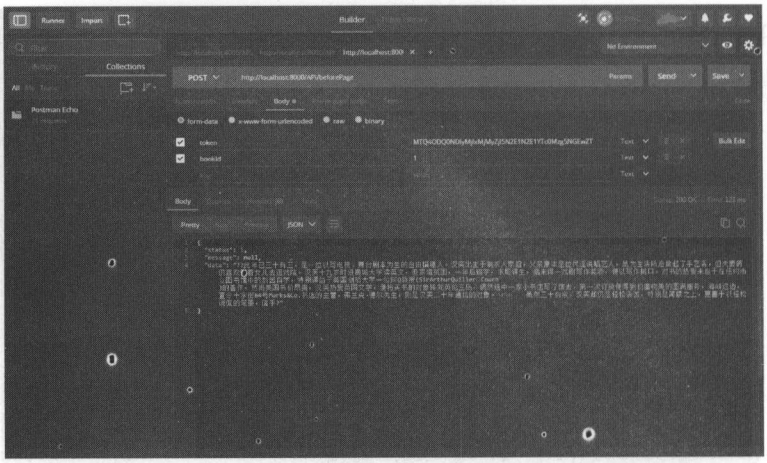


图 7-24　上一页

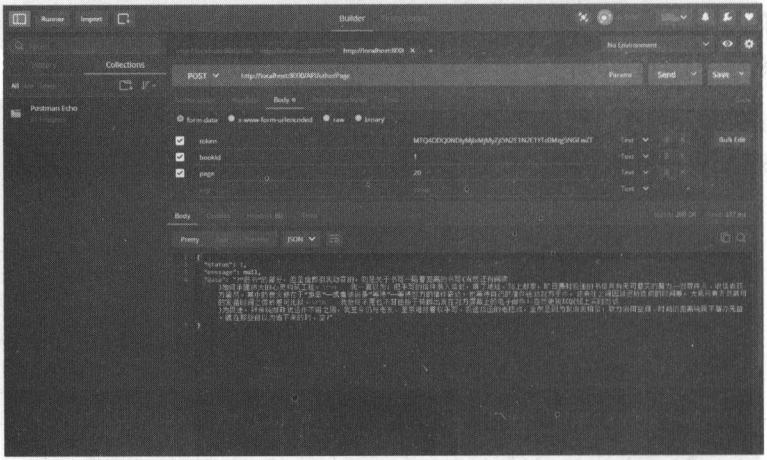


图 7-25　跳页操作

注意：当请求的页数越界后，将会返回一个错误，并且将页数重置，所以无须担心发生未知错误。

至此，所有后端服务器的开发已经基本完成，并且测试成功，下一节将会进入小程序的开发中。

7.3　项目微信小程序端编码

完成了上面的操作后，本系统已经完成了一大半，鉴于前面的知识，估计本节对读者来说会比较简单，为了方便读者的学习，这里依然给出详细的步骤。

7.3.1 微信小程序页面设计

本例的微信小程序需要实现的基本功能有：

- 用户的注册和登录。
- 所有书籍信息的获取。
- 历史阅读书的获取。
- 书籍的阅读和上下页的翻阅。
- 书籍的跳页。

所以，对于小程序需要实现的页面有：

- 用户的注册页面。
- 用户的登录页面。
- 用户登入后的书籍主页。
- 用户登入后的历史阅读。
- 用户阅读页面（包括翻页按钮和跳页的输入框）。

注意：本例程序不再特别考虑 UI 和相关页面样式，只以实现功能为主。

7.3.2 搭建微信小程序项目

从本节开始进行正式的小程序编码工作。

注意：首先需要将测试服务器启动，并且保证其运行没有问题，通过浏览器可以正常访问。

建立新的小程序项目，如图 7-26 所示。

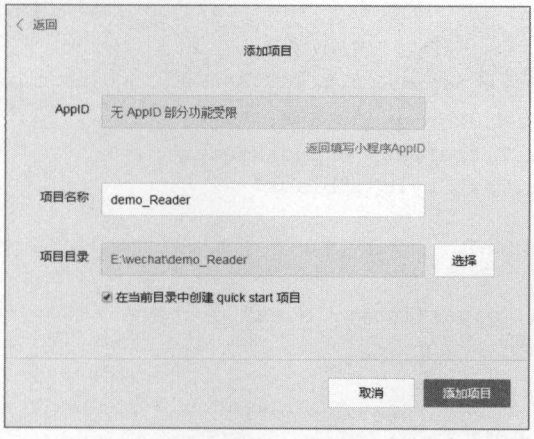


图 7-26 新建项目

需要根据上一节中的页面设计规划相关的路由，页面路由通过app.json文件配置，具体的路由规划如下所示：

```
"pages":[
    "pages/index/index",
    "pages/logs/logs",
    "pages/user/login",
    "pages/user/register",
    "pages/user/index",
    "pages/book/read"
  ],
```

按Ctrl+s组合键保存此页面路由代码，此时小程序编辑器会自动生成相应的目录和文件，如图7-27所示。

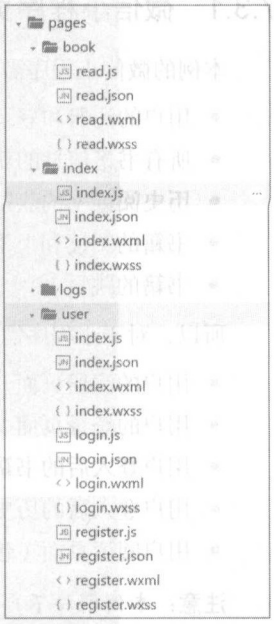


图7-27　项目目录和文件

获取所有的书本信息无须登录，可以制作一个进入小程序直接获取到书本信息的应用，只有需要查看用户历史或阅读书本时才需要登录。

可以通过配置app.json中的菜单和其他选择来进行小程序页面的配置，完整的app.json代码如下所示：

```
{
  "pages":[
    "pages/index/index",
    "pages/logs/logs",
    "pages/user/login",
    "pages/user/register",
    "pages/user/index",
    "pages/book/read"
  ],
  "window":{
    "backgroundTextStyle":"light",
    "navigationBarBackgroundColor": "#fff",
    "navigationBarTitleText": "BookReader",
    "navigationBarTextStyle":"black"
  },
   "tabBar": {
    "list": [{
      "pagePath": "pages/index/index",
      "text": "首页"
    }, {
      "pagePath": "pages/user/index",
      "text": "用户历史"
```

```
    }]
  }
}
```

这时，小程序的展示效果如图 7-28 所示。

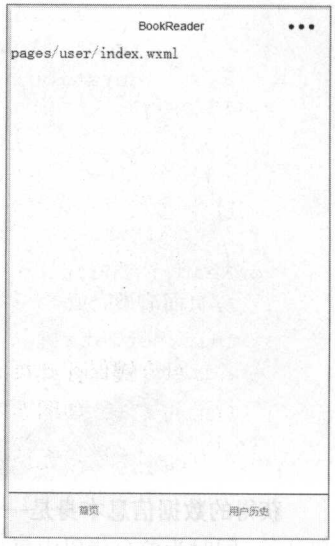


图 7-28 小程序的展示效果

7.3.3 微信小程序“首页”菜单的代码实现

打开 index/index.js 文件，我们在此文件中编写代码，获得所有的服务器书本信息。

注意：本章使用的并非公网上的服务器。而是本地的环境，所以请求 IP 地址为 127.0.0.1，即本机地址。

启动后台服务器，在 index/index.js 中编写代码。实现从服务器端获取到所有的小说信息，并且进行显示。通过向服务器发起 GET 请求进行数据的获取，代码如下所示：

```
//index.js
//获取应用实例
var app = getApp()
var temp_data;
Page({
  data: {
    items:[]
  },
  onLoad: function (e) {
    console.log(1)
    //页面初始化后发出新的请求
    wx.request({
      url: 'http://localhost:8000/API', //仅为示例，并非真实的接口地址
      data: {
      },
      method: 'GET',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        console.log(res.data.data)
        if (res.data.status == 1) {
          temp_data = res.data.data
        } else {
          wx.showToast({
```

```
            title: res.data.message,
            icon: 'loading',
            duration: 2000
          })
        }
      }
    })
  },
  onReady: function () {
    //页面渲染完成
    this.setData({
      items:temp_data
    })
  },
})
```

获得的数据信息本身是一个 json 对象，所以直接对其绑定数据即可显示，进行前台的数据绑定和显示。同时为前台代码中每一本小说增加一个链接，通过单击此链接可以进入文章的具体阅读页，通过文章的 id 进行文章的识别。前台页面的代码如下所示：

```
<!--index.wxml-->
<view class="container">
  <view class="list" wx:for="{{items}}" wx:key="id">
    <navigator                              url="../book/read?id={{item.id}}"
hover-class="navigator-hover">
      <text class="article_title">{{item.name}}</text>
      <text class="article_username">作者: {{item.writer}}</text>
      <view>
        <text class="article_date">描述: {{item.description}}</text>
      </view>
    </navigator>
  </view>
</view>
```

需要对其样式进行细节上的调整，其 wxss 代码如下所示。

```
/**index.wxss**/
.list{
   height:230px;
   width: 100%;
   border-bottom: 1px solid #000;
}
.article_title{
```

```
    padding: 10px;
    font-size: 20px;
}
.article_username{
    font-size: 13px;
}
.article_date{
    padding: 10px;
    font-size: 10px;
    color: gray
}
.btn{
    width: 50%;
    font-size:15px;
}
```

Index 主页的显示效果如图 7-29 所示。

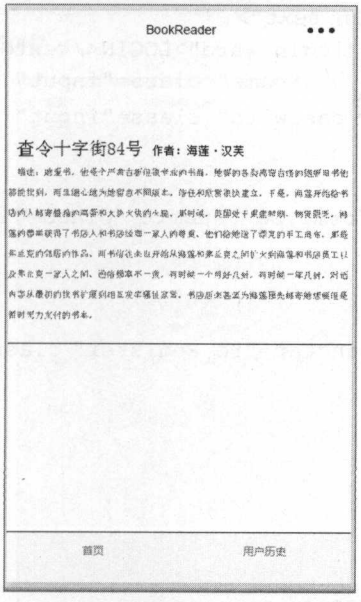


图 7-29 index 主页的显示效果

7.3.4 微信小程序“用户历史”菜单的代码实现

单击“用户历史”菜单时涉及页面逻辑，在用户未登录的时候，不能直接让用户进入此页面，在未登录的情况下进入此页面，会自动跳转至一个登录页面。所以需要在 user/index.js 中新增代码。

```
onLoad:function(options){
    //页面初始化 options 为页面跳转所带来的参数
    console.log(wx.getStorageSync('usertoken'))
    if(wx.getStorageSync('usertoken')==''){
      wx.navigateTo({
          url: './login'
        })
    }
  }
```

通过检测是否存在已经登录后的 userToken，如果不存在此缓存，则直接跳转至登录的路由页面。

新增登录路由页面和逻辑。对于登录的页面设计为以下页面代码，可以仿照第 5 章介绍过的日记登录页面。

```
<!--pages/user/login.wxml-->
<view class="main">
    <form bindsubmit="formSubmit" bindreset="formReset">
        <view class="main_text">
            <text class="login_word">LOGIN</text>
            <input name="username" class="input" placeholder="用户名"/>
            <input name="password" class="input" password placeholder="密码
"/>
            <view>
                <button formType="submit" class="login_btn" size="mini">登录
</button>
            </view>
            <view>
                <button bindtap="re_Register" class="login_btn" size="mini">
注册</button>
            </view>
        </view>
    </form>
</view>
```

对样式进行调整：

```
/* pages/user/login.wxss */
.main{
    background: #99ffff;
    width: 100%;
    height:598px;
}
.main_text{
```

```
    padding-top: 80px;
    padding-left: 40%;
}
.login_word{
    color: #ffffff;
    font-size: 30px;
}
.input{
    background: #ffffff;
    margin-left: -20%;
    margin-top: 10%;
    width: 80%;
}
.login_btn{
    margin-top:10%;
    margin-left: 3%;
}
```

登录页面的显示效果如图 7-30 所示。

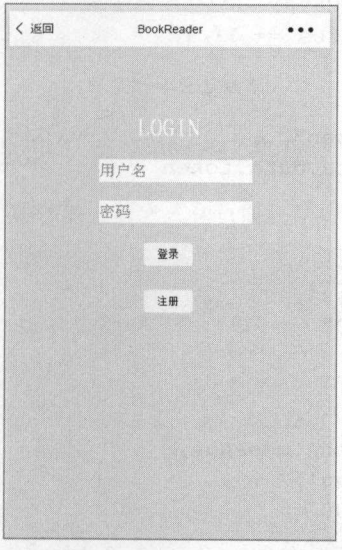


图 7-30 登录页面的显示效果

页面基本的逻辑操作就是收集用户名和密码，并将其发送给服务器端进行验证，根据返回值判断是否登录成功，不成功时返回登录错误信息，否则返回登录成功的信息。登录 login.js 代码如下：

```
//pages/user/login.js
Page({
```

```
  re_Register: function (e) {
    wx.navigateTo({
      url: './register',
    })
  },
  formSubmit: function (e) {
    //当单击了 submit 按钮后发出新的请求
    wx.request({
      url: 'http://localhost:8000/API/login', //仅为示例，并非真实的接口地址
      data: {
        'username': e.detail.value.username,
        'password': e.detail.value.password,
      },
      method: 'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        console.log(res.data)
        if (res.data.status == 1) {
          //存入缓存中
          wx.setStorage({
            key: "usertoken",
            data: res.data.data.token
          })
          wx.showToast({
            title: '登录成功',
            icon: 'success',
            duration: 2000
          })
        } else {
          wx.showToast({
            title: res.data.message,
            icon: 'loading',
            duration: 2000
          })
        }
      }
    })
  },
  data: {},
})
```

如果输入信息正确，则会返回登录成功，并且在缓存中记录 userToken 值，登录成功如图 7-31 所示。



图 7-31 登录成功

7.3.5 微信小程序“用户注册”页面的代码实现

用户注册页面可以仿照日记小程序的注册方式和页面。本页的 wxml 代码如下：

```
<!--pages/user/register.wxml-->
<view class="main">
    <form bindsubmit="formSubmit" bindreset="formReset">
        <view class="main_text">
            <text class="login_word">REGISTER</text>
            <input name="username" class="input" placeholder="用户名"/>
            <input name="password" class="input" password placeholder="密码
"/>
            <input name="re_password" class="input" password placeholder="
重新输入密码"/>
            <view>
                <button formType="submit" class="login_btn" size="mini">注册
</button>
            </view>
        </view>
```

```
    </form>
</view>
```

wxss 样式代码如下：

```
/* pages/user/register.wxss */
.main{
    background: #99ffff;
    width: 100%;
    height:598px;
}
.main_text{
    padding-top: 80px;
    padding-left: 40%;
}
.login_word{
    color: #ffffff;
    font-size: 30px;
}
.input{
    background: #ffffff;
    margin-left: -20%;
    margin-top: 10%;
    width: 80%;
}
.login_btn{
    margin-top:10%;
    margin-left: 3%;
}
```

完整的页面逻辑如下：

```
//pages/user/register.js
Page({
  data: {},
  formSubmit: function (e) {
    //对于是否两次密码输入一致进行判断
    if (e.detail.value.password == e.detail.value.re_password) {
      //当单击了 submit 按钮后发出新的请求
      wx.request({
        url: 'http://localhost:8000/API/register',//仅为示例，并非真实的接口地址
        data: {
          'username': e.detail.value.username,
          'password': e.detail.value.password,
```

```
        },
        method: 'POST',
        header: {
          'content-type': 'application/x-www-form-urlencoded'
        },
        success: function (res) {
          console.log(res.data)
          if (res.data.status == 1) {
            wx.showToast({
              title: res.data.message+",请返回登录。",
              icon: 'success',
              duration: 2000
            })
          } else {
            wx.showToast({
              title: res.data.message,
              icon: 'loading',
              duration: 2000
            })
          }
        }
      })
    } else {
      wx.showToast({
        title: "两次输入密码不一致，请检查",
        icon: 'loading',
        duration: 2000
      })
    }
  },
})
```

注意：这里为了方便开发，仅仅增加了用户名和密码两个选项。

注册页面的显示效果如图 7-32 所示。

返回到用户主页中，此时，不应当简单地检测是否存在用户的 userToken，而是需要对其 Token 进行验证，并且重新验证其是否需要登录。如果登录成功则可以进行用户书签的请求，如果不成功，则不请求直接显示为空的页面即可。

注意：当然，这里可以设计得更加人性化一些，如显示不同的提示和更人性化的跳转等。

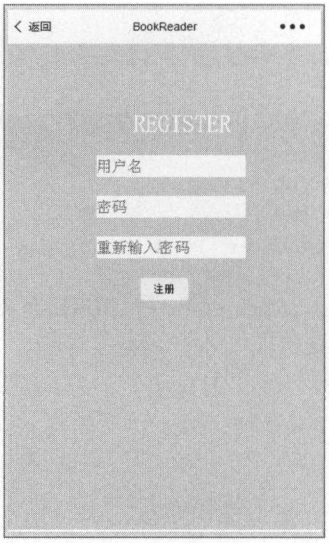


图 7-32　注册页面的显示效果

更新 user/index.js 逻辑代码如下所示：

```
//pages/user/index.js
var temp_data;
Page({
  data: {
    items:[]
  },
  onLoad: function (options) {
    //页面初始化 options 为页面跳转所带来的参数
    console.log(wx.getStorageSync('usertoken'))
    if (wx.getStorageSync('usertoken') == '') {
      wx.navigateTo({
        url: './login'
      })
    } else {
      //页面初始化发送请求，验证是否登录
      wx.request({
        url: 'http://localhost:8000/API/login', //仅为示例，并非真实的接口地址
        data: {
          'token': wx.getStorageSync('usertoken')
        },
        method: 'POST',
        header: {
```

```
            'content-type': 'application/x-www-form-urlencoded'
          },
          success: function (res) {
            //console.log(wx.getStorageSync('usertoken'))
            console.log(res.data)
            if (res.data.status == 1) {
              //页面初始化发送请求，获取用户列表
              wx.request({
                url: 'http://localhost:8000/API/books', //仅为示例，并非真实的接
口地址
                data: {
                  'token': wx.getStorageSync('usertoken')
                },
                method: 'POST',
                header: {
                  'content-type': 'application/x-www-form-urlencoded'
                },
                success: function (res) {
                  console.log(res.data.data[0])
                  if (res.data.status == 1) {
                    console.log(res.data.data[0])
                    temp_data = res.data.data[0]
                  } else {
                    wx.showToast({
                      title: res.data.message,
                      icon: 'loading',
                      duration: 1000
                    })
                  }
                }
              })
            } else {
              wx.showToast({
                title: res.data.message,
                icon: 'loading',
                duration: 1000,
                success: function (res) {
                  wx.redirectTo({
                    url: './login'
                  })
                }
```

```
            })
          }
        }
      })
    }
  },
  onReady: function () {
    //页面渲染完成
    this.setData({
      items: temp_data
    })
    console.log(items)
  },
  onShow: function () {
    //页面显示
  },
  onHide: function () {
    //页面隐藏
  },
  onUnload: function () {
    //页面关闭
  }
})
```

显示界面和主页显示方法相同，页面代码如下所示：

```
<!--pages/user/index.wxml-->
<text>用户历史页</text>
<view class="container">
  <view class="list" wx:for="{{items}}" wx:key="id">
    <navigator                              url="../book/read?id={{item.id}}"
hover-class="navigator-hover">
      <text class="article_title">{{item.name}}</text>
      <text class="article_username">作者：{{item.writer}}</text>
      <view>
        <text class="article_date">描述：{{item.description}}</text>
      </view>
    </navigator>
  </view>
</view>
```

样式代码如下所示：

```
.list{
```

```
    height:230px;
    width: 100%;
    border-bottom: 1px solid #000;
}
.article_title{
    padding: 10px;
    font-size: 20px;
}
.article_username{
    font-size: 13px;
}
.article_date{
    padding: 10px;
    font-size: 10px;
    color: gray
}
```

用户历史页的效果如图 7-33 所示。

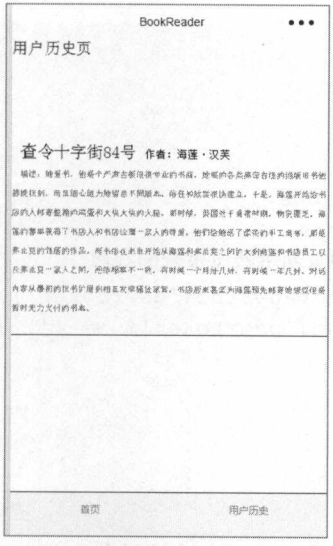


图 7-33 用户历史页的显示效果

7.3.6 微信小程序“小说阅读”页面的代码实现

最重要的是小说阅读页面，其路由为 book/read。通过单击书籍可以跳转至此页面。此页面也需要一个简单的用户验证，其操作和请求本身需要通过用户登录后的 Token 实现，所以如果用户未登录则跳转至登录页面。同时，此页面需要两个按钮，用作上一页和下一页的链接。

小说阅读页面的前台显示代码如下：

```
<!--pages/book/read.wxml-->
<text>{{items}}</text>
<view>
    <button class="btn" style='float:left' bindtap="beforePage">上一页
</button>
    <button class="btn" type="default" bindtap="nextPage">下一页</button>
</view>
```

样式代码如下：

```
/* pages/book/read.wxss */
.btn{
    width: 50%;
    font-size:15px;
}
```

页面逻辑代码如下：

```
//pages/book/read.js
var temp_data;
var temp_options;
Page({
  data: {},
  onLoad: function (options) {
    temp_options = options
    //页面初始化 options 为页面跳转所带来的参数
    console.log(wx.getStorageSync('usertoken'))
    if (wx.getStorageSync('usertoken') == '') {
      wx.navigateTo({
        url: './login'
      })
    } else {
      wx.request({
        //下一页或者初始化
        url: 'http://localhost:8000/API/nextPage', //仅为示例，并非真实的接口
地址
        data: {
          'token': wx.getStorageSync('usertoken'),
          'bookId': options.id
        },
        method: 'POST',
        header: {
          'content-type': 'application/x-www-form-urlencoded'
```

```
        },
        success: function (res) {
          console.log(res.data.data[0])
          if (res.data.status == 1) {
            console.log(res.data.data)
            temp_data = res.data.data
          } else {
            wx.showToast({
              title: res.data.message,
              icon: 'loading',
              duration: 1000
            })
          }
        }
      })
    }
  },
  onReady: function () {
    //页面渲染完成
    this.setData({
      items: temp_data
    })
  },
  onShow: function () {
    //页面显示
  },
  onHide: function () {
    //页面隐藏
  },
  onUnload: function () {
    //页面关闭
  },
  beforePage: function (e) {
    wx.request({
      //上一页
      url: 'http://localhost:8000/API/beforePage', //仅为示例，并非真实的接口
地址
      data: {
        'token': wx.getStorageSync('usertoken'),
        'bookId': temp_options.id
      },
      method: 'POST',
```

```
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        console.log(res.data.data[0])
        if (res.data.status == 1) {
          console.log(res.data.data)
          temp_data = res.data.data

        } else {
          wx.showToast({
            title: res.data.message,
            icon: 'loading',
            duration: 1000
          })
        }
      }
    })
    this.setData({
      items: temp_data
    })
  },
  nextPage: function (e) {
    wx.request({
      //下一页或者初始化
      url: 'http://localhost:8000/API/nextPage',//仅为示例，并非真实的接口地址
      data: {
        'token': wx.getStorageSync('usertoken'),
        'bookId': temp_options.id
      },
      method: 'POST',
      header: {
        'content-type': 'application/x-www-form-urlencoded'
      },
      success: function (res) {
        console.log(res.data.data[0])
        if (res.data.status == 1) {
          console.log(res.data.data)
          temp_data = res.data.data

        } else {
          wx.showToast({
            title: res.data.message,
```

```
            icon: 'loading',
            duration: 1000
          })
        }
      }
    })
    this.setData({
      items: temp_data
    })
  }
})
```

书籍阅读页面的显示效果如图 7-34 所示。



图 7-34　书籍阅读页面的显示效果

7.4　小结

本章通过一个完整的实例，从零开始进行了一个小程序应用的搭建。本章不再拘泥于小程序本身，而是从服务器端入手，通过真正完全可运行的服务器和小程序的交互，介绍了小程序本身的生态和开发流程。

本例通过 PHP 端服务器的介绍，扩宽了开发者的视野，让读者更容易体会到其交互过程和应用的本质。

通过对于本章的学习，读者可以认识到：

- 服务器端开发的基本方式和流程。
- 服务器端和小程序端基本的交互方式和流程。
- 更加了解小程序的开发。
- 更加了解一个真实开发的环境和环节。

7.5　实践与练习

通过学习本章的内容，读者不仅需要对本章介绍的相关内容进行练习和模仿，如果对于后端开发拥有一定的兴趣，而且需要更加深入地学习和练习。

8 小程序新增 API

截至 2017 年 3 月 27 日，面对开发者和使用者小程序开放了更多的权限，其中包括了支持绝大部分的 ES 6 API，新增全新的 API，如蓝牙、卡券、收货地址等接口。这些新功能使得微信小程序变得更加强大和丰富，也为小程序的广泛应用打下了基础。

本章主要涉及的知识点有：

- 蓝牙相关的 API。
- 其他更新 API。

8.1 蓝牙 API

对于支持蓝牙连接的设备而言，其支持的形式一般有两种：

- 普通的经典蓝牙。
- 低功耗蓝牙。

所以微信小程序提供了两种不同的 API 用于蓝牙连接。

低功耗蓝牙采用的模式和连接方式与普通蓝牙不同，BLE（Bluetooh Low Energy）是蓝牙低功耗的简称。BLE 技术是低成本、短距离、可互操作的鲁棒性无线技术，工作在免许可的 2.4GHz ISM 射频频段。它从一开始就设计为超低功耗（ULP）无线技术，可利用许多智能手段最大限度地降低功耗。

8.1.1 获取蓝牙适配器

1. 经典蓝牙

wx.openBluetoothAdapter(OBJECT)，用于初始化蓝牙适配器，负责开启蓝牙和初始化蓝牙操作，参数说明如表 8-1 所示。

表 8-1 初始化蓝牙适配器参数

参 数	类 型	必 填	说 明
success	Function	是	成功则返回成功初始化信息
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 8-1】

初始化蓝牙适配器：

```
wx.openBluetoothAdapter({
  success: function (res) {
    console.log(res)
  }
})
```

注意：由于系统问题，目前仅在 Mac 版的开发工具上支持蓝牙调试，基础库版本 1.1.0 开始支持，低版本需做兼容处理。iOS 微信客户端 6.5.6 版本开始支持，Android 客户端暂不支持。

2. 低功耗蓝牙

wx.createBLEConnection(OBJECT)，此 API 用于连接低功耗蓝牙设备，OBJECT 参数如表 8-2 所示。

表 8-2 OBJECT参数说明

参 数	类 型	必 填	说 明
deviceId	string	是	蓝牙设备 id，参考 getDevices 接口
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 8-2】

连接低功耗蓝牙设备：

```
wx.createBLEConnection({
  //这里的 deviceId 需要在上面的 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取
  deviceId: deviceId,
  success: function (res) {
    console.log(res)
  }
})
```

8.1.2 关闭蓝牙模块

1. 经典蓝牙

wx.closeBluetoothAdapter(OBJECT)，用于对蓝牙模块的关闭操作，调用该方法将断开所有已建立的连接并释放系统资源，具体参数说明如表 8-3 所示。

表 8-3 关闭蓝牙模块参数

参　数	类　型	必　填	说　明
success	Function	是	成功则返回成功关闭模块信息
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 8-3】

经典蓝牙：

```
wx.closeBluetoothAdapter({
  success: function (res) {
    console.log(res)
  }
})
```

2. 低功耗蓝牙

wx.closeBLEConnection(OBJECT)，此 API 用于断开与低功耗蓝牙设备的连接，OBJECT 参数说明如表 8-4 所示。

表 8-4 OBJECT参数说明

参　数	类　型	必　填	说　明
deviceId	String	是	蓝牙设备 id，参考 getDevices 接口
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 8-4】

关闭低功耗蓝牙：

```
wx.closeBLEConnection({
  success: function (res) {
    console.log(res)
  }
})
```

8.1.3 获取本机蓝牙状态

1. 经典蓝牙

wx.getBluetoothAdapterState(OBJECT)，用于获取本机蓝牙适配器状态，请求参数说明如表 8-5 所示。

表 8-5 请求参数说明

参 数	类 型	必 填	说 明
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

请求成功时会返回不同的参数如表 8-6 所示。

表 8-6 请求成功时返回参数说明

参 数	类 型	说 明
adapterState	Object	蓝牙适配器信息
errMsg	String	成功：ok；错误：详细信息

蓝牙适配器信息 adapterState，同样拥有两种不同的参数，其参数如表 8-7 所示。

表 8-7 蓝牙适配器信息参数

参 数	类 型	说 明
discovering	Boolean	是否正在搜索设备
available	Boolean	蓝牙适配器是否可用

【示例 8-5】

获取本机蓝牙适配器状态：

```
wx.getBluetoothAdapterState({
  success: function (res) {
    console.log(res)
  }
})
```

wx.onBluetoothAdapterStateChange(CALLBACK)，此 API 用于监听蓝牙适配器状态变化事件，其回调参数如表 8-8 所示。

【示例 8-6】

监听蓝牙适配器状态：

```
wx.onBluetoothAdapterStateChange(function(res) {
  console.log(`adapterState changed, now is`, res)
})
```

表 8-8　回调参数

参　数	类　型	说　明
available	Boolean	蓝牙适配器是否可用
discovering	Boolean	蓝牙适配器是否处于搜索状态

2. 低功耗蓝牙

wx.onBLEConnectionStateChanged(CALLBACK)，此 API 用于监听低功耗蓝牙连接的错误事件，包括设备丢失、连接异常断开等，具体的回调参数如表 8-9 所示。

表 8-9　回调参数

参　数	类　型	说　明
deviceId	String	蓝牙设备 id，参考 device 对象
connected	Boolean	连接目前的状态

【示例 8-7】

监听低功耗蓝牙连接：

```
wx.onBLEConnectionStateChanged(function(res) {
  console.log(`device ${res.deviceId} state has changed, connected:
${res.connected}`)
})
```

wx.getBLEDeviceServices(OBJECT)，此 API 用于获取低功耗蓝牙设备所有 service（服务），具体的参数说明如表 8-10 所示。

表 8-10　OBJECT参数说明

参　数	类　型	必　填	说　明
deviceId	String	是	蓝牙设备 id，参考 getDevices 接口
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

如果请求成功，则返回 success 的相关参数，其参数说明如表 8-11 所示。

成功获取蓝牙设备服务信息的具体参数参，如表 8-12 所示。

表 8-11 success返回参数

参 数	类 型	说 明
services	Array	设备服务列表
errMsg	String	成功：ok；错误：详细信息

表 8-12 服务信息的参数

参 数	类 型	说 明
uuid	String	蓝牙设备服务的 uuid
isPrimary	Boolean	该服务是否为主服务

【示例 8-8】

获取低功耗蓝牙适配器状态：

```
wx.getBLEDeviceServices({
  //这里的 deviceId 需要在 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取
  deviceId: deviceId,
  success: function (res) {
    console.log('device services:', res.services)
  }
})
```

8.1.4 搜索蓝牙设备

1. 经典蓝牙

wx.startBluetoothDevicesDiscovery(OBJECT)，此 API 用于已经获取蓝牙设备后，开始搜寻附近的蓝牙外围设备，其参数说明如表 8-13 所示。

注意：该操作比较耗费系统资源，请在搜索并连接到设备后调用 stop 方法停止搜索。

表 8-13 OBJECT参数说明

参 数	类 型	必 填	说 明
services	Array	否	蓝牙设备主 service 的 uuid 列表
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

注意：services 参数，某些蓝牙设备会广播自己的主 service 的 uuid。如果这里传入该数组，那么根据该 uuid 列表，只搜索有这个主 service 的设备。

success 返回参数如表 8-14 所示。

表 8-14　success返回参数

参　数	类　型	说　明
errMsg	String	成功：ok；错误：详细信息

【示例 8-9】

搜索经典蓝牙：

```
wx.startBluetoothDevicesDiscovery({
  success: function (res) {
    console.log(res)
  }
})
```

wx.stopBluetoothDevicesDiscovery(OBJECT)，此 API 用于在搜索蓝牙设备时，停止搜寻附近的蓝牙外围设备。请在确保找到需要连接的设备后调用该方法停止搜索。其参数说明如表 8-15 所示。

表 8-15　OBJECT参数说明

参　数	类　型	必　填	说　明
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数如表 8-16 所示。

表 8-16　success返回参数

参　数	类　型	说　明
errMsg	String	成功：ok；错误：详细信息

adapterState 返回的蓝牙适配器状态信息说明如表 8-17 所示。

表 8-17　返回的蓝牙适配器状态信息

参　数	类　型	说　明
discovering	Boolean	是否正在搜索设备
available	Boolean	蓝牙适配器是否可用

【示例 8-10】

停止搜寻附近的蓝牙外围设备：

```
wx.stopBluetoothDevicesDiscovery({
```

```
  success: function (res) {
    console.log(res)
  }
})
```

wx.getBluetoothDevices(OBJECT)，此 API 用于获取所有已发现的蓝牙设备，包括已经和本机处于连接状态的设备，其参数说明如表 8-18 所示。

表 8-18 OBJECT参数说明

参 数	类 型	必 填	说 明
services	Array	否	蓝牙设备主 service 的 uuid 列表
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数如表 8-19 所示。

表 8-19 success返回参数

参 数	类 型	说 明
devices	Array	uuid 对应的已连接设备列表
errMsg	string	成功：ok；错误：详细信息

device 对象获得的设备连接列表的参数说明如表 8-20 所示。

表 8-20 设备连接列表的参数说明

参 数	类 型	说 明
name	String	蓝牙设备名称，某些设备可能没有
deviceId	String	用于区分设备的 id
RSSI	Int	当前蓝牙设备的信号强度
advertisData	ArrayBuffer	当前蓝牙设备的广播内容

注意：开发者工具和 Android 上获取到的 deviceId 为设备 MAC 地址，iOS 上则为设备 uuid。

【示例 8-11】

获取所有已发现的蓝牙设备：

```
wx.getBluetoothDevices({
  success: function (res) {
    console.log(res)
  }
})
```

wx.onBluetoothDeviceFound(CALLBACK)，此 API 用于监听寻找到新设备的事件，对于其回调函数的说明如表 8-21 所示。

8-21　CALLBACK回调函数的说明

参　数	类　型	说　明
devices	Array	新搜索到的设备列表

【示例 8-12】

监听新设备：

```
wx.onBluetoothDeviceFound(function(devices) {
  console.log('new device list has founded')
  console.dir(devices)
})
```

wx.getConnectedBluetoothDevices(OBJECT)，此 API 可以根据 uuid 获取处于已连接状态的设备，其参数说明如表 8-22 所示。

表 8-22　OBJECT参数说明

参　数	类　型	必　填	说　明
services	Array	是	蓝牙设备主 service 的 uuid 列表
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

对于其请求 success 返回参数如表 8-23 所示。

表 8-23　success返回参数

参　数	类　型	说　明
devices	Array	搜索到的设备列表
errMsg	string	成功：ok；错误：详细信息

device 对象，获得的蓝牙设备信息如表 8-24 所示。

表 8-24　蓝牙设备信息

参　数	类　型	说　明
name	String	蓝牙设备名称，某些设备可能没有
deviceId	String	用于区分设备的 id

注意：开发者工具和 Android 上获取到的 deviceId 为设备 MAC 地址，iOS 上则为设备 uuid。

【示例 8-13】

获取已连接的设备：

```
wx.getConnectedBluetoothDevices({
  success: function (res) {
    console.log(res)
  }
})
```

2. 低功耗蓝牙

wx.getBLEDeviceCharacteristics(OBJECT)，此 API 用于获取低功耗蓝牙设备所有 characteristic（特征值），其参数说明如表 8-25 所示。

表 8-25 OBJECT参数说明

参 数	类 型	必 填	说 明
deviceId	String	是	蓝牙设备 id，参考 device 对象
serviceId	String	是	蓝牙服务 uuid
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数如表 8-26 所示。

表 8-26 success返回参数

参 数	类 型	说 明
characteristics	Array	设备特征值列表
errMsg	String	成功：ok；错误：详细信息

请求成功后返回的特征值（characteristic）对象的参数如表 8-27 所示。

表 8-27 蓝牙设备的特征值对象的参数

参 数	类 型	说 明
uuid	String	蓝牙设备特征值的 uuid
properties	Object	该特征值支持的操作类型

对于其中的 properties 对象的参数如表 8-28 所示。

表 8-28 properties对象的参数

参 数	类 型	说 明
read	Boolean	该特征值是否支持 read 操作

续表

参　数	类　型	说　明
write	Boolean	该特征值是否支持 write 操作
notify	Boolean	该特征值是否支持 notify 操作
indicate	Boolean	该特征值是否支持 indicate 操作

【示例 8-14】

搜索低功耗蓝牙设备：

```
wx.getBLEDeviceCharacteristics({
  //这里的 deviceId 需要在 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取
  deviceId: deviceId,
  //这里的 serviceId 需要在 getBLEDeviceServices 接口中获取
  serviceId: serviceId,
  success: function (res) {
    console.log('device getBLEDeviceCharacteristics:', res.characteristics)
  }
})
```

wx.readBLECharacteristicValue(OBJECT)，此 API 用于读取低功耗蓝牙设备的特征值的二进制数据值，其参数说明如表 8-29 所示。

注意：设备的特征值必须支持 read 才可以成功调用，具体参照特征值的 properties 属性。

表 8-29　OBJECT参数说明

参　数	类　型	必　填	说　明
deviceId	String	是	蓝牙设备 id，参考 device 对象
serviceId	String	是	蓝牙特征值对应服务的 uuid
characteristicId	String	是	蓝牙特征值的 uuid
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

对于其 success 的返回参数如表 8-30 所示。

表 8-30　success返回参数

参　数	类　型	说　明
characteristic	Object	设备特征值信息
errMsg	String	成功：ok；错误：详细信息

对于其中的特征值对象的参数如表 8-31 所示。

表 8-31 蓝牙设备的特征值对象的参数

参 数	类 型	说 明
characteristicId	String	蓝牙设备特征值的 uuid
serviceId	Object	蓝牙设备特征值对应服务的 uuid
value	ArrayBuffer	蓝牙设备特征值对应的二进制值

【示例 8-15】

读取低功耗蓝牙设备的特征值：

```
//必须在这里回调才能获取
wx.onBLECharacteristicValueChange(function(characteristic) {
  console.log('characteristic value comed:', characteristic)
})

wx.readBLECharacteristicValue({
  //这里的 deviceId 需要在 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取
  deviceId: deviceId,
  //这里的 serviceId 需要在 getBLEDeviceServices 接口中获取
  serviceId: serviceId,
  //这里的 characteristicId 需要在 getBLEDeviceCharacteristics 接口中获取
  characteristicId: characteristicId,
  success: function (res) {
    console.log('readBLECharacteristicValue:', res.characteristic.value)
  }
})
```

注意：并行调用多次读写接口存在读写失败的可能性。read 接口读取到的信息需要在 onBLECharacteristicValueChange 方法注册的回调中获取。

wx.writeBLECharacteristicValue(OBJECT)，此 API 用于向低功耗蓝牙设备特征值中写入二进制数据，其具体的参数说明如表 8-32 所示。

注意：设备的特征值必须支持 write 才可以成功调用，具体参照特征值的 properties 属性。

表 8-32 OBJECT参数说明

参 数	类 型	必 填	说 明
deviceId	String	是	蓝牙设备 id，参考 device 对象
serviceId	String	是	蓝牙特征值对应服务的 uuid

续表

参　数	类　型	必　填	说　明
characteristicId	String	是	蓝牙特征值的 uuid
value	ArrayBuffer	是	蓝牙设备特征值对应的二进制值
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

对于请求成功时返回的 success 的具体参数如表 8-33 所示。

表 8-33　success返回参数

参　数	类　型	说　明
errMsg	String	成功：ok；错误：详细信息

【示例 8-16】

写入低功耗蓝牙设备的特征值：

```
//这里的回调可以获取到 write 导致的特征值改变
wx.onBLECharacteristicValueChange(function(characteristic) {
  console.log('characteristic value changed:', characteristic)
})

wx.writeBLECharacteristicValue({
  //这里的 deviceId 需要在 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取
  deviceId: deviceId,
  //这里的 serviceId 需要在 getBLEDeviceServices 接口中获取
  serviceId: serviceId,
  //这里的 characteristicId 需要 getBLEDeviceCharacteristics 接口中获取
  characteristicId: characteristicId,
  success: function (res) {
    console.log('writeBLECharacteristicValue success', res.errMsg)
  }
})
```

wx.notifyBLECharacteristicValueChanged(OBJECT)，此 API 作用于启用低功耗蓝牙设备特征值变化时的 notify 功能，其具体的参数说明如表 8-34 所示。

注意：设备的特征值必须支持 notify 才可以成功调用，具体参照 characteristic 的 properties 属性，必须先启用 notify 才能监听到设备 characteristicValueChange 事件。

表 8-34 OBJECT参数说明

参 数	类 型	必 填	说 明
deviceId	String	是	蓝牙设备 id，参考 device 对象
serviceId	String	是	蓝牙特征值对应服务的 uuid
characteristicId	String	是	蓝牙特征值的 uuid
state	Boolean	是	true：启用 notify；false：停用 notify
success	Function	是	成功则返回本机蓝牙适配器状态
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

对于请求成功时返回的 success 的具体参数如表 8-35 所示。

表 8-35 success返回参数

参 数	类 型	说 明
errMsg	String	成功：ok，错误：详细信息

【示例 8-17】

启用低功耗蓝牙设备的 notify 功能：

```
wx.notifyBLECharacteristicValueChanged({
  state: true, //启用 notify 功能
  //这里的 deviceId 需要在 getBluetoothDevices 或 onBluetoothDeviceFound 接口中获取
  deviceId: deviceId,
  //这里的 serviceId 需要在 getBLEDeviceServices 接口中获取
  serviceId: serviceId,
  //这里的 characteristicId 需要在 getBLEDeviceCharacteristics 接口中获取
  characteristicId: characteristicId,
  success: function (res) {
    console.log('notifyBLECharacteristicValueChanged success', res.errMsg)
  }
})
```

wx.onBLEConnectionStateChanged(CALLBACK)，此 API 监听低功耗蓝牙连接的错误事件，包括设备丢失、连接异常断开等，其回调参数如表 8-36 所示。

表 8-36 CALLBACK的回调参数

参 数	类 型	说 明
deviceId	String	蓝牙设备 id，参考 device 对象
connected	Boolean	连接目前的状态

【示例 8-18】

监听低功耗蓝牙连接的错误事件：

```
wx.onBLEConnectionStateChanged(function(res) {
  //该方法回调中可以用于处理连接意外断开等异常情况
  console.log(`device ${res.deviceId} state has changed, connected: ${res.connected}`)
})
```

wx.onBLECharacteristicValueChange(CALLBACK)，它用于监听低功耗蓝牙设备的特征值变化。必须先启用 notify 接口才能接收到设备推送的 notification，其回调参数如表 8-37 所示。

表 8-37 CALLBACK的回调参数

参 数	类 型	说 明
deviceId	String	蓝牙设备 id，参考 device 对象
serviceId	String	特征值所属服务 uuid
characteristicId	String	特征值 uuid
value	ArrayBuffer	特征值最新的值

【示例 8-19】

监听低功耗蓝牙设备的特征值变化：

```
wx.onBLECharacteristicValueChange(function(res) {
  console.log(`characteristic ${res.characteristicId} has changed, now is ${res.value}`)
})
```

8.1.5 上手实践：蓝牙管理器

本节将会采用一个蓝牙管理器的完整实例进行蓝牙 API 的说明和讲解。

1. 新建项目 demo_bluetooth

对于此项目，需要设置多个蓝牙事件的监听按钮，所以前台界面的代码如下：

```
<!--index.wxml-->
<button type="default" bindtap="bt_init" class="btn_css"> 初始化蓝牙适配器 </button>
<button type="default" bindtap="bt_get"class="btn_css"> 获取本机蓝牙适配器状态 </button>
<button type="default" bindtap="bt_find"class="btn_css"> 搜寻附近的蓝牙外围设备 </button>
```

```
    <button type="default" bindtap="bt_endfind"class="btn_css"> 停止搜寻附近的蓝牙外围设备 </button>
    <button type="default" bindtap="bt_getfound"class="btn_css"> 获取所有已发现的蓝牙设备 </button>
    <button type="default" bindtap="bt_getLinked"class="btn_css"> 获取处于已连接状态的设备 </button>
    <button type="default" bindtap="bt_close"class="btn_css"> 关闭蓝牙 </button>
```

为了使其样式变得更加整洁，为其添加简单的长度限制和间距，具体的 wxss 代码如下：

```
.btn_css {
 margin-top: 10px;
 width: 90%;
}
```

蓝牙管理器的显示效果如图 8-1 所示。

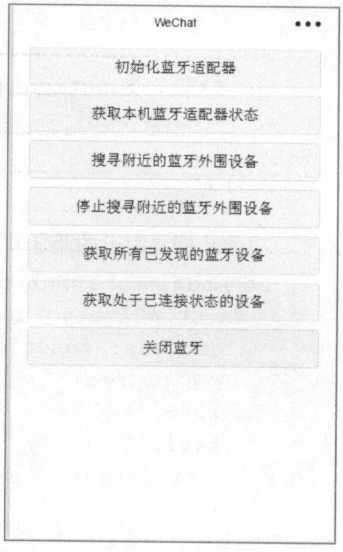


图 8-1 蓝牙管理器的显示效果

2. 监听事件和页面逻辑

监听事件的代码如下：

```
//index.js
//获取应用实例
var app = getApp()
Page({
  data: {
  },
  //初始化蓝牙函数
  bt_init: function () {
    wx.openBluetoothAdapter({
      success: function (res) {
        tips(res)
      },
      fail: function (res) {
        tips(res)
      }
    })
  },
  //获取本机蓝牙适配器
  bt_get: function () {
    wx.getBluetoothAdapterState({
      success: function (res) {
```

```
        tips(res)
      },
      fail: function (res) {
        tips(res)
      }
    })
  },
  //搜寻附近的蓝牙外围设备
  bt_find: function () {
    wx.startBluetoothDevicesDiscovery({
      success: function (res) {
        tips(res)
      },
      fail: function (res) {
        tips(res)
      }
    })
  },
  //停止搜寻附近的蓝牙外围设备
  bt_endfind: function () {
    wx.stopBluetoothDevicesDiscovery({
      success: function (res) {
        tips(res)
      },
      fail: function (res) {
        tips(res)
      }
    })
  },
  //获取所有已发现的蓝牙设备
  bt_getfound: function () {
    wx.getBluetoothDevices({
      success: function (res) {
        tips(res)
      },
      fail: function (res) {
        tips(res)
      }
    })
  },
  //获取处于已连接状态的设备
```

```
  bt_getLinked: function () {
    wx.getConnectedBluetoothDevices({
      success: function (res) {
        tips(res)
      },
      fail: function (res) {
        tips(res)
      }
    })
  },
  //关闭蓝牙
  bt_close: function () {
    wx.closeBluetoothAdapter({
      success: function (res) {
        tips(res)
      },
      fail: function (res) {
        tips(res)
      }
    })
  }
})
//弹出提示框
function tips(temp_res) {
  wx.showToast({
    title: JSON.stringify(temp_res),
    icon: 'success',
    duration: 2000
  })
}
```

其效果为单击相关的蓝牙操作按钮，弹出提示框，显示当前是否成功地调用小程序的功能和其中包含的信息。

8.2 其他新增 API

微信的目的是更方便小程序的应用，所以微信的很多功能都对小程序进行了开放，如地址、卡券等，本节介绍如何实现这些很常见的小功能。

8.2.1 获取收货地址

wx.chooseAddress(OBJECT)，用于调用用户编辑收货地址原生界面，并在编辑完成后返回用户选择的地址，其参数说明如表 8-38 所示。

注意：基础库版本 1.1.0 开始支持，低版本需做兼容处理，微信客户端 6.5.6 版本开始支持。

表 8-38 OBJECT参数说明

参 数	类 型	必 填	返回
success	Function	否	返回用户选择的收货地址信息
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

对于请求成功时返回的 success 的具体参数如表 8-39 所示。

表 8-39 success返回参数

参 数	类 型	说 明
errMsg	String	调用结果
userName	String	收货人姓名
postalCode	String	邮编
provinceName	String	国标收货地址第一级地址
cityName	String	国标收货地址第二级地址
countyName	String	国标收货地址第三级地址
detailInfo	String	详细收货地址信息
nationalCode	String	收货地址国家码
telNumber	String	收货人手机号码

【示例 8-20】

获取收货地址：

```
wx.chooseAddress({
  success: function (res) {
    console.log(res.userName)
    console.log(res.postalCode)
    console.log(res.provinceName)
    console.log(res.cityName)
    console.log(res.countyName)
    console.log(res.detailInfo)
    console.log(res.nationalCode)
    console.log(res.telNumber)
```

```
    }
})
```

获取用户地址如图 8-2 所示。

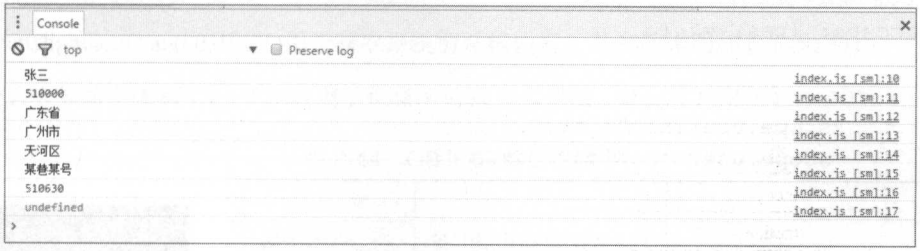


图 8-2 获得用户地址

8.2.2 打开小程序设置界面

wx.openSetting(OBJECT)，此 API 用于调起客户端小程序设置界面，返回用户设置的操作结果，其参数说明如表 8-40 所示。

注意：从基础库版本 1.1.0 开始支持，低版本需做兼容处理。微信客户端从 6.5.6 版本开始支持。

表 8-40 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	否	接口调用成功的回调函数，返回内容详见返回参数说明。
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

对于请求成功时返回的 success 的具体参数如表 8-41 所示。

表 8-41 success返回参数

参 数	类 型	说 明
authSetting	Object	用户授权结果，其中 key 为 scope 值，value 为 bool 值，表示用户是否允许授权

对于返回 scope 值的说明和对应的接口如表 8-42 所示。

表 8-42 scope值

scope	对应接口
scope.userInfo	wx.getUserInfo
scope.userLocation	wx.getLocation, wx.chooseLocation
scope.address	wx.chooseAddress
scope.record	wx.startRecord

【示例 8-21】

打开小程序的设置界面：

```
wx.openSetting({
  success: (res) => {
    /*
     * res = {
     *   "scope.userInfo": true,
     *   "scope.userLocation": true
     * }
    /*
  }
})
```

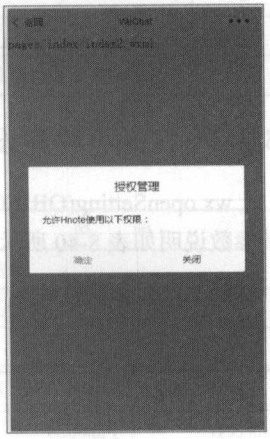


图 8-3　获取权限的显示效果

注意：在调试工具上无法进行权限的获取。

获取权限的显示效果如图 8-3 所示。

8.2.3　操作剪贴板

wx.setClipboardData(OBJECT)用于设置系统剪贴板的内容，其参数说明如表 8-43 所示。

注意：从基础库版本 1.1.0 开始支持，低版本需做兼容处理。微信客户端从 6.5.6 版本开始支持。

表 8-43　OBJECT参数说明

参　数	类　型	必　填	说　明
data	String	是	需要设置的内容
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 8-22】

设置系统剪贴板：

```
wx.setClipboardData({
  data: 'data',
  success: function(res) {
    wx.getClipboardData({
      success: function(res) {
        console.log(res.data) //data
```

```
        }
      })
    }
  })
```

wx.getClipboardData(OBJECT)用于获取系统剪贴板内容，其参数说明如表 8-44 所示。

注意：从基础库版本 1.1.0 开始支持，低版本需做兼容处理，微信客户端从 6.5.6 版本开始支持。

表 8-44 OBJECT参数说明

参 数	类 型	必 填	说 明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

对于请求成功时 success 返回具体参数如表 8-45 所示。

表 8-45 success返回参数

参 数	类 型	说 明
data	String	剪贴板的内容

【示例 8-23】

获取系统剪贴板：

```
wx.getClipboardData({
  success: function(res){
    console.log(res.data)
  }
})
```

作为开发工具的测试，结果会在控制台中打印，如图 8-4 所示。

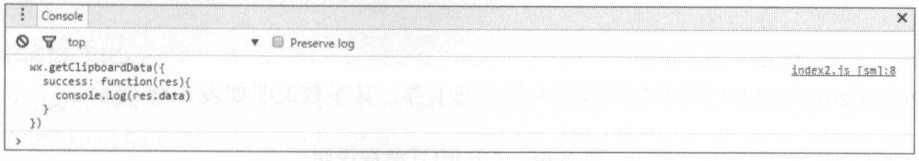


图 8-4 获取剪贴板的测试结果

8.2.4 支持微信卡券

wx.addCard(OBJECT)用于批量添加卡券，其参数说明如表 8-46 所示。

注意：从基础库版本 1.1.0 开始支持，低版本需做兼容处理。微信客户端从 6.5.4 版本开始支持。

表 8-46　OBJECT参数说明

参　数	类　型	必　填	说　明
cardList	ObjectArray	是	需要添加的卡券列表
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

回调结果如表 8-47 所示。

表 8-47　回调结果

回调类型	errMsg	说　明
success	addCard:ok	添加卡券成功
fail	addCard:fail cancel	用户取消添加卡券
fail	addCard:fail(detail message)	添加卡券失败，其中 detail message 为后台返回详细的失败原因

【示例 8-24】

批量添加卡券：

```
wx.addCard({
  cardList: [
    {
      cardId: '',
      cardExt: '{"code": "", "openid": "", "timestamp": "", "signature":""}'
    }, {
      cardId: '',
      cardExt: '{"code": "", "openid": "", "timestamp": "", "signature":""}'
    }
  ],
  success: function(res) {
    console.log(res.cardList)                    //卡券添加结果
  }
})
```

wx.openCard(OBJECT)用于查看微信卡包中的卡券，其参数说明如表 8-48 所示。

表 8-48　OBJECT参数说明

参　数	类　型	必　填	说　明
cardList	ObjectArray	是	需要打开的卡券列表
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

【示例 8-25】

查看微信卡包中的卡券：

```
wx.openCard({
  cardList: [
    {
      cardId: '',
      code: ''
    }, {
      cardId: '',
      code: ''
    }
  ],
  success: function(res) {
  }
})
```

注意：目前开发者工具上尚未支持小程序卡券的调试，请在真机上调试。

8.2.5 监听网络变化

wx.onNetworkStatusChange(CALLBACK)是网络的补充 API，用于监听网络环境状态的变化，其回调的参数如表 8-49 所示。

注意：从基础库版本 1.1.0 开始支持，低版本需做兼容处理。微信客户端从 6.5.6 版本开始支持。

表 8-49 CALLBACK的回调参数

参　数	类　型	说　明
isConnected	Boolean	当前是否有网络连接
networkType	String	网络类型，有效值 Wi-Fi/2g/3g/4g/unknown（Android 下不常见的网络类型）/none（无网络）

【示例 8-26】

监听网络变化：

```
wx.onNetworkStatusChange(function(res) {
  console.log(res.isConnected)
  console.log(res.networkType)
})
```

当使用上述代码进行网络切换时，会在控制台打印相关的信息，具体的效果如图 8-5 所示。

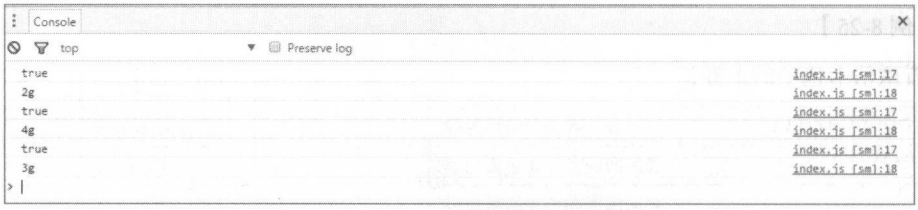


图 8-5　切换网络

8.3　ES 6 的支持

ECMAScript 6（以下简称 ES 6）是 JavaScript 语言的下一代标准。因为 ES 6 是在 2015 年发布的，所以又称 ECMAScript 2015。

也就是说，ES 6 就是 ES 2015。ES 6 在 ES 5 的基础上进行了大篇新特性的增加和修改，使 JavaScript 变得更加强大。

注意：本书并非对于 JavaScript 的介绍学习，且受篇幅所限，本书也不能介绍所有 ES 6 的新特性，所以这里只介绍一些常用的特性。

下面介绍一些 ES 6 的特性。

1．let 语句

let 语句与 var 类似，但是对于 var 定义的为全局变量而言，let 定义后自身会形成一个作用域，在此作用域中其变量发生作用。

```
for (let i = 0; i < 3; i++) {
let i = 'abc';
console.log(i);
}
//abc
//abc
//abc
```

2．const 语句

Const 语句声明一个只读的常量。一旦声明，常量的值就不能改变。

```
const PI = 3.1415;
```

对于 PI 进行赋值操作则会报错。

3．数组的解构赋值

直接来看一段代码：

```
let a = 1;
let b = 2;
let c = 3;
```

它等同于：

```
let [a, b, c] = [1, 2, 3];
```

注意：此解构赋值可用于变量的交换赋值和对 json 对象的解析操作。

4. 字符串方法

- includes()：返回布尔值，表示是否找到了参数字符串。
- startsWith()：返回布尔值，表示参数字符串是否在源字符串的头部。
- endsWith()：返回布尔值，表示参数字符串是否在源字符串的尾部。

5. 模板字符串

模板字符串（template string）是增强版的字符串，用反引号（`）标识。它可以当作普通字符串使用，也可以用来定义多行字符串，或者在字符串中嵌入变量、方法，甚至逻辑运算。

```
//在字符串中嵌入变量
var name = "XXX";
`Hello ${name}`
```

6. 数值方法

ES 6 将全局方法 parseInt()和 parseFloat()移植到 Number 对象上面，行为完全保持不变。

- Number.isFinite()用来检查一个数值是否为有限的（finite）。
- Number.isNaN()用来检查一个值是否为 NaN。

它们与传统的全局方法 isFinite()和 isNaN()的区别在于，传统方法先调用 Number()将非数值的值转为数值，再进行判断，而这两种新方法只对数值有效，非数值一律返回 false。

7. 数组方法

Array.from 方法用于将两类对象转为真正的数组：类似数组的对象（array-like object）和可遍历（iterable）的对象（包括 ES 6 新增的数据结构 Set 和 Map，只要是部署了 Iterator 接口的数据结构，Array.from 都能将其转为数组。Array.from 还可以接受第二个参数，作用类似于数组的 Map 方法，用来对每个元素进行处理，将处理后的值放入返回的数组）。

```
//下面的例子将数组中布尔值为 false 的成员转为 0
Array.from([1, , 2, , 3], (n) => n || 0)
// [1, 0, 2, 0, 3]
```

扩展运算符（...）也可以将某些数据结构转为数组。扩展运算符背后调用的是遍历器接口（Symbol.iterator），如果一个对象没有部署这个接口，就无法转换。

```
//arguments 对象
function foo() {
 var args = [...arguments];
}
//NodeList 对象
[...document.querySelectorAll('div')]
```

8. 对象

ES 6 允许在对象中直接写变量。这时，属性名为变量名，属性值为变量的值。

```
function f(x, y) {
return {x, y};
}
//等同于
function f(x, y) {
return {x: x, y: y};
}
f(1, 2) //Object {x: 1, y: 2}
```

Object.is 方法用于比较两个值是否严格相等，与严格比较运算符（===）的行为基本一致。

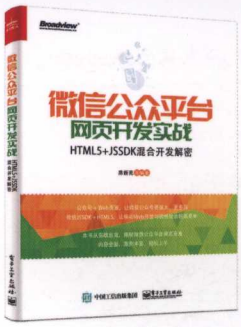
```
Object.is('foo', 'foo')
//true
Object.is({}, {})
//false
```

8.4 小结

本章通过对于小程序相关的技术进行版本的更新，及更多 API 的开放使得整个小程序的开发变得更加丰富，同时也拥有了更好的生态。

8.5 实践与练习

1. 如果有条件可以在 Mac 或者真机上进行蓝牙测试。
2. 结合本章新增 API 和老版本 API，思考其 API 更新的意义。











欢迎反馈意见或投稿
邮箱：dongying@phei.com.cn
电话：010-88254047
微信号：yingzidd

Broadview®
www.broadview.com.cn
博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版

微信小程序开发
零基础入门

涵盖小程序涉及的几乎所有技术

- 基础技术的概念，包括ES 6、MVC等。
- 小程序页面开发：微信IDE、WXSS、WXML等。
- Web页面开发：HTML 5、CSS 3、JavaScript等。
- 微信API开发：多媒体、位置、支付、绘图等。
- 服务器端交互：PHP+MySQL+Laravel+小程序。

博文视点Broadview
新浪微博 weibo.com
@博文视点Broadview

策划编辑：董　英
责任编辑：徐津平
封面设计：吴海燕

上架建议：移动开发>微信

ISBN 978-7-121-31428-5

定价：79.00元